Neural Computing in Quaternion Algebra

by

Toshifumi Minemoto

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Engineering

University of Hyogo, Japan

March 2017

Contents

Chapter 1

Introduction

Recently, the term of 'Artificial Intelligence' frequently appears in various news articles. In Japan, it is predicted that the productive age population will decrease sharply in the future as the population ages and fewer babies are born and it is strongly expected that artificial intelligence as a substitute for labor force and the use of robots will be utilized. It is attracting attention as not only industrial fields with shortage of manpower but also means for creating new value in fields that have been dependent on human experience and intuition, e.g., Fintech such as financial services making full use of information technology and predictive maintenance of factory production facilities in the manufacturing industry.

The driving force behind the expectations for industrial applications of these artificial intelligence, which actually means computational intelligence, is deep learning. Deep learning is a generic term for machine learning methods using multi-layered models of neural networks which is inspired by the information processing in the brain. Machine learning has been developed as a field of artificial intelligence since the end of the 1950s as a research to realize learning abilities equivalent to brains by computer, not by explicit programming. As shown by the success of DeepMind's AlphaGo [1], this deep learning has attracted a great deal of attention in the field of machine learning in recent years due to its high performance.

The growth of deep learning is attribute to the following factors: One is that the

information infrastructure including the Internet has been improved. The global data volume doubles every two years, and it will exceed 40,000 exabytes in 2020 [2]. In particular, it becomes possible to acquire various kinds of information such as images and moving images with the development of social media and the evolution of various sensors, That is, it is easy to collect large amounts of data that can be used for learning as compared with before. Next is a dramatic improvement in the performance of the computer. Due to the significant improvement in computing performance by the evolution of both hardware and software, such as GPUs, multi-core CPUs, large capacity memories, parallel processing by computer clusters, a large-scale neural network model which has been difficult to evaluate has become possible to execute on a computer. Thus, the exponential improvement in computer performance indicated in the performance transition of TOP 500 [3] supercomputer is accelerating research on machine learning. The last and most important is the steady accumulation of long years of solid sober research in neural networks. Deep learning is a neural network models which has various layers at least on the surface. That is, its basic part has not changed much from the conventional neural networks. Various basic algorithms and models for neural networks have been proposed before 1990, e.g., back propagation algorithm enables learning of multi-layered neural networks, and Boltzmann machine is the prototype of deep brief network [4] which is cited as a breakthrough of deep learning. Moreover, Neocognitron [5] and LeNet [6] are the prototype of convolutional neural networks which is frequently used for recent image recognition systems. It can be said that there is the development of deep learning today because of these findings. Several approaches for solving problems which are caused in deep networks has been proposed such as an activation function by rectified linear unit and effective adjustment methods of learning rate for gradient descent learning by ADAM and AdaGrad. They are also the main factor for the achievement of deep learning.

Extension of neural networks on hypercomplex number system is one of such research efforts. Input, output, and internal state of a neuron which is the basic computational unit are represented by hypercomplex number in these types of neural networks.

Complex numbers and quaternions, which are ones of hypercomplex numbers, are useful number systems for various engineering problems. It is well known that complex numbers are employed for signal processing dealing with amplitude and phase information. Quaternions are widely employed in robotics and computer graphics due to the fact that they provide a convenient mathematical notation for representing orientations and rotations of objects in three dimensions. From the viewpoint of information processing, the essential significance of neural networks and other machine learning methods is to acquire information expressions which capture the intrinsic structure inherent in data through learning process. In that sense, it is thought that it is very useful to employ complex numbers and quaternions that can calculate two or three dimensional information as a unit as expressions of neurons. In fact, it is suggested that complex-valued and quaternionic feed forward neural networks have a remarkable learning ability in terms of affine transformation problems in two or three dimensional space in previous studies [7, 8, 9]. Hypercomplex-valued neural networks hold the potential for higher performance than that of the conventional real-valued neural networks. Therefore, introducing complex numbers and quaternions into neural networks is expected to achieve performance improvement of deep learning. Although the number of applications of neural networks employing quaternions is comparatively less than that of complex-valued neural networks.

In this study, we focus on quaternion-based neural computing which can be expected to be effective in future application fields such as robot control and color image processing and recognition. We address several problems existing mutually connected neural networks and multilayered neural networks, and propose methods for solving these problems and aim for gaining a new understanding of quaternionic extensions of the existing methods.

Chapter 2

Quaternion Algebra

2.1 Definitions of Quaternions

In this section, we show the definitions and notations of quaternions. The Quaternion algebra was first described by W. R. Hamilton in 1843 [10]. Quaternions form a class of hypercomplex numbers that consist of a real number and three kinds of imaginary units, $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$. Formally, a quaternion is defined as a vector in a four-dimensional vector space,

$$x = x^{(e)} + x^{(i)}\boldsymbol{i} + x^{(j)}\boldsymbol{j} + x^{(k)}\boldsymbol{k}, \tag{2.1}$$

where $x^{(e)}$, $x^{(i)}$, $x^{(j)}$ and $x^{(k)}$ are real numbers. The multiplications between the three imaginary units obey the following rules:

$$\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1, \quad \boldsymbol{ij} = -\boldsymbol{ji} = \boldsymbol{k}. \tag{2.2}$$

These rules are summarized in The multiplication rules of the quaternions are shown in table 2.1. In this table, the columns on the left-hand side give the first factor while the top row indicates the second factor. This is important since hypercomplex numbers are not commutative.

A quaternion is also written using 4-tuple or 2-tuple notations as follows:

$$x = \left(x^{(e)}, x^{(i)}, x^{(j)}, x^{(k)}\right) = \left(x^{(e)}, \vec{x}\right), \tag{2.3}$$

Table 2.1: Multiplication rules between imaginary units $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$.

	1	$\boldsymbol{i}$	$\boldsymbol{j}$	$\boldsymbol{k}$
1	1	$\boldsymbol{i}$	$\boldsymbol{j}$	$\boldsymbol{k}$
$\boldsymbol{i}$	$\boldsymbol{i}$	-1	$\boldsymbol{k}$	$-\boldsymbol{j}$
$\boldsymbol{j}$	$\boldsymbol{j}$	$-\boldsymbol{k}$	-1	$\boldsymbol{i}$
$\boldsymbol{k}$	$\boldsymbol{k}$	$\boldsymbol{j}$	$-\boldsymbol{i}$	-1

where $\vec{x} = (x^{(i)}, x^{(j)}, x^{(k)})$. In this representation $x^{(e)}$ is the scalar part of x, and $\vec{x}$ forms the vector part.

Now, we show the definition of arithmetic operations between quaternions, p and q. The addition and subtraction of quaternions are defined in the same manner as those of complex numbers or vectors by

$$p \pm q = \left(p^{(e)} \pm q^{(e)}, \vec{p} \pm \vec{q}\right) \tag{2.4}$$

$$= \left(p^{(e)} \pm q^{(e)}, p^{(i)} \pm q^{(i)}, p^{(j)} \pm q^{(j)}, p^{(k)} \pm q^{(k)}\right). \tag{2.5}$$

The product of p and q, denoted as pq, is defined from Eq.(2.2) as

$$pq = \left(p^{(e)}q^{(e)} - \vec{p} \cdot \vec{q}, p^{(e)}\vec{q} + q^{(e)}\vec{p} + \vec{p} \times \vec{q}\right), \tag{2.6}$$

where $\vec{p} \cdot \vec{q}$ and $\vec{p} \times \vec{q}$ denote the dot and cross products respectively between three dimensional vectors $\vec{p}$ and $\vec{q}$. The multiplication between scalar $a = (a, \vec{0})$ and quaternion q is given by

$$aq = \left(aq^{(e)}, a\vec{q}\right) = \left(aq^{(e)}, aq^{(i)}, aq^{(j)}, aq^{(k)}\right). \tag{2.7}$$

The quaternion conjugate is defined as

$$q^* = \left(q^{(e)}, -\vec{q}\right) = q^{(e)} - q^{(i)}\boldsymbol{i} - q^{(j)}\boldsymbol{j} - q^{(k)}\boldsymbol{k}. \tag{2.8}$$

The conjugate of product holds the relation of $(pq)^* = q^*p^*$. The quaternion norm of q, notation $|q|$, is defined by

$$|q| = \sqrt{qq^*} = \sqrt{q^{(e)2} + q^{(i)2} + q^{(j)2} + q^{(k)2}}, \tag{2.9}$$

and the reciprocal of a non-zero quaternion q is defined as

$$q^{-1} = \frac{q^*}{|q|^2}. \tag{2.10}$$

The quaternion involutions, which are self-inverse mappings, are defined [11] as:

$$q^{\boldsymbol{i}} = -\boldsymbol{i}q\boldsymbol{i} = q^{(e)} + q^{(i)}\boldsymbol{i} - q^{(j)}\boldsymbol{j} - q^{(k)}\boldsymbol{k}, \tag{2.11}$$

$$q^{\boldsymbol{j}} = -\boldsymbol{j}q\boldsymbol{j} = q^{(e)} - q^{(i)}\boldsymbol{i} + q^{(j)}\boldsymbol{j} - q^{(k)}\boldsymbol{k}, \tag{2.12}$$

$$q^{\boldsymbol{k}} = -\boldsymbol{k}q\boldsymbol{k} = q^{(e)} - q^{(i)}\boldsymbol{i} - q^{(j)}\boldsymbol{j} + q^{(k)}\boldsymbol{k}. \tag{2.13}$$

By using these involutions, four components of a quaternion can be represented as:

$$q^{(e)} = \frac{1}{4}\left(q + q^{\boldsymbol{i}} + q^{\boldsymbol{j}} + q^{\boldsymbol{k}}\right), \tag{2.14}$$

$$q^{(i)} = \frac{1}{4\boldsymbol{i}}\left(q + q^{\boldsymbol{i}} - q^{\boldsymbol{j}} - q^{\boldsymbol{k}}\right), \tag{2.15}$$

$$q^{(j)} = \frac{1}{4\boldsymbol{j}}\left(q - q^{\boldsymbol{i}} + q^{\boldsymbol{j}} - q^{\boldsymbol{k}}\right), \tag{2.16}$$

$$q^{(k)} = \frac{1}{4\boldsymbol{k}}\left(q - q^{\boldsymbol{i}} - q^{\boldsymbol{j}} + q^{\boldsymbol{k}}\right). \tag{2.17}$$

A quaternion and its conjugate can be also expressed as a linear function of the involutions as follows:

$$q = \frac{1}{2}\left(q^{\boldsymbol{i}*} + q^{\boldsymbol{j}*} + q^{\boldsymbol{k}*} - q^*\right), \tag{2.18}$$

$$q^* = \frac{1}{2}\left(q^{\boldsymbol{i}} + q^{\boldsymbol{j}} + q^{\boldsymbol{k}} - q\right). \tag{2.19}$$

2.2 Quaternions in polar representation

A complex value $c = c^{(e)} + \boldsymbol{i}c^{(i)}$ in Cartesian form can also be represented as one in polar form: $c = r \cdot \mathrm{e}^{\boldsymbol{i}\theta}$, where $r = \sqrt{c^{(e)2} + c^{(i)2}}$ and $\theta = \tan^{-1} c^{(i)}/c^{(e)}$. In a similar way, a quaternion x in Cartesian form can be transformed into one in a polar representation. We adopt the representation defined by Bülow [12, 13] in this study. Quaternion x can be represented in the polar form

$$x = |x|\mathrm{e}^{\boldsymbol{i}\varphi}\mathrm{e}^{\boldsymbol{k}\psi}\mathrm{e}^{\boldsymbol{j}\theta}, \tag{2.20}$$

where

$$\mathrm{e}^{\boldsymbol{i}\varphi} = \cos\varphi + \boldsymbol{i}\sin\varphi, \quad \mathrm{e}^{\boldsymbol{j}\theta} = \cos\theta + \boldsymbol{j}\sin\theta, \quad \mathrm{e}^{\boldsymbol{k}\psi} = \cos\psi + \boldsymbol{k}\sin\psi. \tag{2.21}$$

The three kinds of angular phase φ, θ, and ψ are defined within the following interval:

$$-\pi \leq \varphi < \pi, \quad -\frac{\pi}{2} \leq \theta < \frac{\pi}{2}, \quad -\frac{\pi}{4} \leq \psi \leq \frac{\pi}{4}. \tag{2.22}$$

The formulas for calculating these phases from the quaternion in Cartesian form are given as Algorithm 1.

begin

$\psi \leftarrow -\frac{\arcsin(2(q^{(i)}q^{(j)} - q^{(e)}q^{(k)}))}{2}$

if $\psi = \pm\frac{\pi}{4}$ **then**

$\varphi \leftarrow 0, \quad \theta \leftarrow \frac{\arg_j(q^{k*}q)}{2}$

else

$\varphi \leftarrow \frac{\arg_i(qq^{j*})}{2}, \quad \theta \leftarrow \frac{\arg_j(q^{i*}q)}{2}$

end

if $\mathrm{e}^{\boldsymbol{i}\varphi}\mathrm{e}^{\boldsymbol{k}\psi}\mathrm{e}^{\boldsymbol{j}\theta} = -q$ **then**

if $\varphi \geq 0$ **then**

$\varphi \leftarrow \varphi - \pi$

else

$\varphi \leftarrow \varphi + \pi$

end

end

end

Algorithm 1: Calculate φ, θ, ψ for a quaternion $x = a + b\boldsymbol{i} + c\boldsymbol{j} + d\boldsymbol{k}$, $|x| = 1$

Chapter 3

Associative Memories in Quaternionic Neural Networks

Hopfield network is an associative memory model based on fully connected neural networks in which information is embedded as connection weights between the neurons. Quaternionic Hopfield associative memory is expected to process three-dimensional multilevel values such as intensities of RGB color components or body coordinates in three dimensional space. The state of a neuron is represented by three kinds of phases in distinct complex planes in the associative memory. However, the performance and properties of such the quaternionic extended models are not clarified.

We evaluate the performance of quaternionic Hopfield associative memory such as storage performance of two learning rules, i.e., Hebbian learning rule and projection learning rule. We also examine noise robustness of the stored memory patterns to evaluate the recall performance. Then, we propose another types of quaternionic associative memory models for improving the performance of the quaternionic Hopfield associative memory. One is quaternionic bipartite auto-associative memory which has two types of neuron layers. The other is quaternionic Hopfield associative memory with dual connections which are utilizing non-commutative property of quaternions. We show these models have superior performance compared to that of the quaternionic Hopfield associative memory through several experiments.

3.1 Introduction

Extensions in complex and hypercomplex number systems on Hopfield-type associative memories (HAMs) [14] have been extensively investigated. Representations for states in the networks are enriched due to the degrees of freedom in these systems. Complex-valued extension of Hopfield Associative memories, called CHAMs, were proposed [15, 16, 17, 18] and their learning schemes for embedding the patterns in the networks were also investigated [19, 20, 21]. Various CHAMs adopt the representation for the state of a neuron as a discrete point in the complex plane, so multilevel values, such as the intensity of a pixel in the gray-scaled image, can be naturally represented in these networks.

One difficulty in CHAMs is the existence of rotated patterns in the network by degenerated states of embedded patterns. When a pattern is to be embedded to a CHAM network where a neuron takes K of quantization levels, $K-1$ rotated patterns are also to be embedded in the network. These rotated patterns makes their mixture patterns, and they become spurious patterns in the network, leading to reduce the noise robustness in retrieving patterns from the network. One possible solution to cope with rotated patterns is to compose a heterogeneous network where real-valued neurons are incorporated in the complex or quaternionic networks. A pattern in the real-valued HAM has only one rotated pattern in which each of elements takes the inverted value. A combination of real-valued and complex-valued/quaternionic patterns makes fewer rotated patterns. Such networks are presented and investigated as Complex-valued Bipartite Auto-Associative Memory (CBAAM) [22].

A quaternionic extension of HAMs is called QHAMs where the state of a neuron is represented by three kinds of phases in distinct complex planes [23]. QHAMs are expected to process three-dimensional multilevel values such as intensities of RGB color components or body coordinates in three dimensional space. QHAMs adopt a similar principle in representing the patterns in the network to that in CHAMs, thus they also suffer from the rotated and mixture patterns. However, the detailed performance of

QHAM is not provided in the previous studies.

In this chapter, we first evaluate the performance of QHAM such as storage performance of two learning rules, i.e., Hebbian learning rule and projection learning rule. We also examine noise robustness of the stored memory patterns to evaluate the recall performance of QHAM. Then, we propose another types of quaternionic associative memory models for improving the performance of QHAMs. One is Quaternionic Bipartite Auto-Associative Memory (QBAAM) and the other is Quaternionic Hopfield Associative Memory with Dual Connections (QHAMDC). QBAAM is a quaternionic extended model of CBAAM. The network architecture of QBAAM has two layers where one layer is composed of quaternionic neurons and the other layer is composed of real-valued neurons. QHAMDC is an another approach for improving the performance of QHAM by utilizing non-commutative property of quaternions.

This chapter is organized as follows. In Section 3.2, we first recapitulate the QHAM model and its learning algorithm. QBAAM and QHAMDC are described in Section 3.3 and Section 3.4, respectively. In Section 3.5, Several experiments results are given in Section 3.5 for comparing the learning algorithms and recall performance. We finish with conclusions in Section 3.6.

3.2 Quaternionic Hopfield Associative Memory

We assume a quaternionic Hopfield associative memory (QHAM) with N quaternionic multistate neurons. All neurons are connected to each other in QHAM as shown in Fig. 3.1. The state of a p-th neuron in QHAM is represented in the polar form described in Sec. 2.2,

$$u_p = \mathrm{e}^{\boldsymbol{i}\varphi_p}\mathrm{e}^{\boldsymbol{k}\psi_p}\mathrm{e}^{\boldsymbol{j}\theta_p}, \tag{3.1}$$

where $|u_p| = 1$. The internal state of the p-th neuron at a discrete time t is defined as

$$h_p(t) = \sum_{q=1}^{N} w_{pq}u_q(t) = \sum_{q=1}^{N} w_{pq}\mathrm{e}^{\boldsymbol{i}\varphi_q(t)}\mathrm{e}^{\boldsymbol{k}\psi_q(t)}\mathrm{e}^{\boldsymbol{j}\theta_q(t)}, \tag{3.2}$$

where $w_{pq} \in \mathbb{H}$ denotes the connection weight from the p-th neuron to the q-th neuron.

We use qsign function which consists of three types of complex-valued multistate signum function as an activation function for quaternionic multistate neurons. Thus, the state of a neuron p at time t is updated as

$$u_p(t+1) = \mathrm{qsign}\left(h_p(t)\right), \tag{3.3}$$

where

$$\begin{aligned} \mathrm{qsign}\left(h_p\right) &= \mathrm{qsign}\left(\mathrm{e}^{\boldsymbol{i}\varphi_p}\,\mathrm{e}^{\boldsymbol{k}\psi_p}\,\mathrm{e}^{\boldsymbol{j}\theta_p}\right) && (3.4)\\ &= \mathrm{csign}_A\left(\mathrm{e}^{\boldsymbol{i}\varphi_p}\right)\,\mathrm{csign}_B\left(\mathrm{e}^{\boldsymbol{k}\psi_p}\right)\mathrm{csign}_C\left(\mathrm{e}^{\boldsymbol{j}\theta_p}\right). && (3.5)\end{aligned}$$

The function csign_A is used for updating φ, and it is defined as

$$\mathrm{csign}_A\left(\mathrm{e}^{\boldsymbol{i}\varphi}\right) = \begin{cases} q_0^{(\varphi)} & \text{for } -\pi \le \arg\left(\mathrm{e}^{\boldsymbol{i}\varphi}\right) < -\pi + \varphi_0 \\ q_1^{(\varphi)} & \text{for } -\pi + \varphi_0 \le \arg\left(\mathrm{e}^{\boldsymbol{i}\varphi}\right) < -\pi + 2\varphi_0 \\ \qquad\vdots & \\ q_{A-1}^{(\varphi)} & \text{for } -\pi + (A-1)\varphi_0 \le \arg\left(\mathrm{e}^{\boldsymbol{i}\varphi}\right) < -\pi + A\varphi_0 \end{cases}, \tag{3.6}$$

where A is the quantization level for φ, and φ_0 denotes a quantization unit which is defined by $\varphi_0 = 2\pi/A$. $q_a^{(\varphi)}$ is a distinct point on a unit circle which is defined as

$$q_a^{(\varphi)} = \exp\left(\boldsymbol{i}\left(-\pi + a\varphi_0 + \frac{\varphi_0}{2}\right)\right), \quad a = 0, \ldots, A-1. \tag{3.7}$$

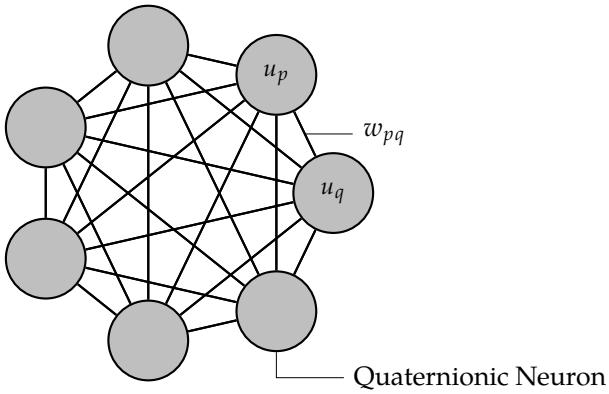


Figure 3.1: Network structure of QHAM.

Therefore the function csign_A outputs the closest quaternion in $\{q_0^{(\varphi)}, \ldots, q_{A-1}^{(\varphi)}\}$ corresponding to the input. Similarly, the function csign_B for updating ψ and the function csign_C for updating θ are defined as follows:

$$\mathrm{csign}_B\left(\mathrm{e}^{\boldsymbol{k}\psi}\right) = \begin{cases} q_0^{(\psi)} & \text{for } -\frac{\pi}{4} \le \arg\left(\mathrm{e}^{\boldsymbol{k}\psi}\right) < -\frac{\pi}{4} + \psi_0 \\ q_1^{(\psi)} & \text{for } -\frac{\pi}{4} + \psi_0 \le \arg\left(\mathrm{e}^{\boldsymbol{k}\psi}\right) < -\frac{\pi}{4} + 2\psi_0 \\ & \vdots \\ q_{B-1}^{(\psi)} & \text{for } -\frac{\pi}{4} + (B-1)\psi_0 \le \arg\left(\mathrm{e}^{\boldsymbol{k}\psi}\right) < -\frac{\pi}{4} + B\psi_0 \end{cases}, \tag{3.8}$$

$$\mathrm{csign}_C\left(\mathrm{e}^{\boldsymbol{j}\theta}\right) = \begin{cases} q_0^{(\theta)} & \text{for } -\frac{\pi}{2} \le \arg\left(\mathrm{e}^{\boldsymbol{j}\theta}\right) < -\frac{\pi}{2} + \theta_0 \\ q_1^{(\theta)} & \text{for } -\frac{\pi}{2} + \theta_0 \le \arg\left(\mathrm{e}^{\boldsymbol{j}\theta}\right) < -\frac{\pi}{2} + 2\theta_0 \\ & \vdots \\ q_{C-1}^{(\theta)} & \text{for } -\frac{\pi}{2} + (C-1)\theta_0 \le \arg\left(\mathrm{e}^{\boldsymbol{j}\theta}\right) < -\frac{\pi}{2} + C\theta_0 \end{cases}, \tag{3.9}$$

where B and C are the quantization levels for φ and θ, respectively. The quantization units ψ_0 and θ_0 are defined by $\psi_0 = \pi/2B$ and $\theta_0 = \pi/C$, respectively. $q_b^{(\psi)}$ and $q_c^{(\theta)}$ are also defined as follows:

$$q_b^{(\psi)} = \exp\left(\boldsymbol{j}\left(-\frac{\pi}{4} + b\psi_0 + \frac{\psi_0}{2}\right)\right), \quad b = 0, \ldots, B-1, \tag{3.10}$$

$$q_c^{(\theta)} = \exp\left(\boldsymbol{k}\left(-\frac{\pi}{2} + c\theta_0 + \frac{\theta_0}{2}\right)\right), \quad c = 0, \ldots, C-1. \tag{3.11}$$

Hence, a quaternionic neuron takes a total of $A \times B \times C$ states. An example of the quantized output points of the quaternionic neuron is shown in Fig. 3.2, where $A = 4$, $B = 2$, and $C = 3$.

The energy function of QHAM takes real value when the connection weights w_{pq} satisfy the following conditions:

$$w_{pq} = w_{qp}^*, \quad w_{pp} = w_{pp}^* = (w^{(e)}, \vec{0}), \quad w^{(e)} \ge 0. \tag{3.12}$$

The function monotonically decreases under the condition $|\Delta\varphi| < \varphi_0, |\Delta\varphi| < \psi_0, |\Delta\psi| < \varphi_0$. $\Delta\varphi$, $\Delta\psi$, $\Delta\theta$ are a phase difference between the state at time $t+1$ and the internal state at time t for the neuron undergoing its update [23].

Two types of learning rules for QHAM are described. Let $\boldsymbol{\xi}^p = (\xi_1^p, \ldots, \xi_N^p)^{\mathrm{T}}$ be a p-th memory pattern ($p = 1, \ldots, P$), where $\xi_i^p \in \{q_a^{(\varphi)} q_b^{(\psi)} q_c^{(\theta)}\}$ and P is the number of embedded memory patterns. Then, the training pattern matrix $\boldsymbol{X}$ is represented as

$$\boldsymbol{X} = (\boldsymbol{\xi}^1, \boldsymbol{\xi}^2, \ldots, \boldsymbol{\xi}^p). \tag{3.13}$$

The connection weight matrix $\boldsymbol{W}$,whose element at (p, q) is denoted as w_{pq}, is given by Hebbian learning rule as:

$$\boldsymbol{W} = \boldsymbol{X}\boldsymbol{X}^* - \mathrm{diag}(\boldsymbol{X}\boldsymbol{X}^*) \tag{3.14}$$

where $\boldsymbol{X}^*$ denotes the conjugate transpose of $\boldsymbol{X}$, and $\mathrm{diag}(\boldsymbol{A})$ is the diagonal matrix whose diagonal elements equal to the diagonal of $\boldsymbol{A}$. This learning rule is simple and fast, however it is difficult to achieve effective storage capacity when memory patterns are not orthogonal to each other.

Next, we describe the projection learning rule, which is a learning algorithm that can embed non-orthogonal (correlated) memory patterns in a network [24]. A key idea of the projection rule is that non-orthogonal patterns are first projected onto orthogonal ones by using inverse matrix, and then the Hebbian rule is applied to these projected patterns. The connection weight matrix $\boldsymbol{W}$ is given by the projection rule as follows:

$$\boldsymbol{W} = \boldsymbol{A} - \mathrm{diag}(\boldsymbol{A}), \quad \boldsymbol{A} = \boldsymbol{X}(\boldsymbol{X}^*\boldsymbol{X})^{-1}\boldsymbol{X}^*, \tag{3.15}$$

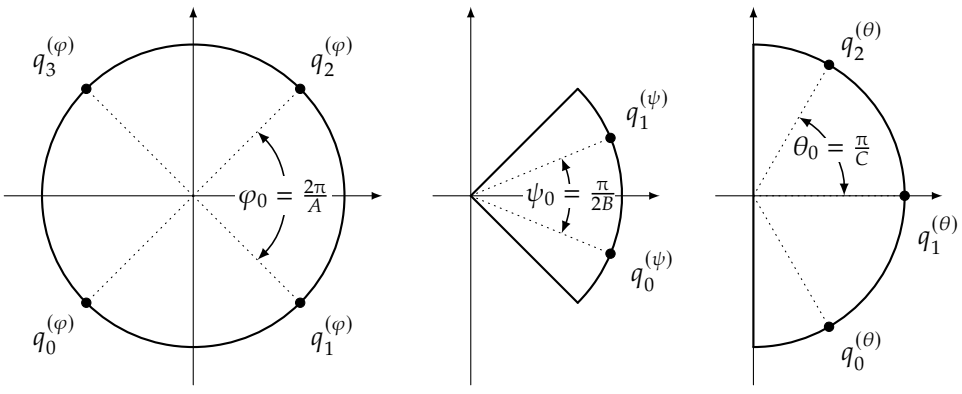


Figure 3.2: An example of quantized output points in the quaternionic multistate neuron($A = 4, B = 2, C = 3$)

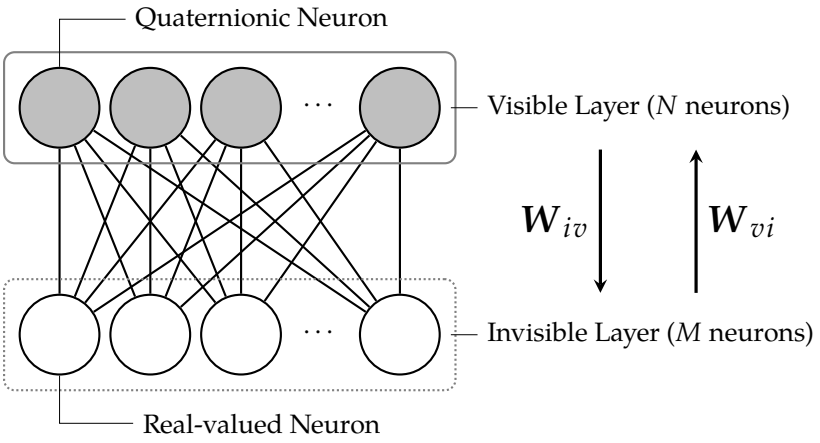


Figure 3.3: Network structure of QBAAM.

The number of patterns that can be stored in the network equals to the number of neurons in the network by using the projection rule.

3.3 Quaternionic Bipartite Auto-Associative Memory

The model of Quaternionic Bipartite Auto-Associative Memory (QBAAM) is described in this section. QBAAM is a quaternionic extension of Complex-valued Bipartite Auto-Associative Memory (CBAAM) [22], in which quaternionic multistate neurons, defined in section 3.2, are used in the network. QBAAM has two layers called visible layer and invisible layer, as shown in Fig. 3.3. The visible layer contains quaternionic multistate neurons, and the invisible layer contains real-valued neurons, i.e., the state of real-valued neurons is +1 or −1. The connection establishes neurons between layers, encoded by quaternionic values, and there are no connections between neurons in each layer. The neurons' outputs in the visible layer are quaternionic values and the connection weights from visible to invisible layers are also quaternionic values, the real-valued neurons in the invisible layers only takes real parts of these quaternionic values. The visible layer can process 3-tuple of multilevel signals, and the recall of rotated patterns of memory patterns can be suppressed by utilizing the neurons in the invisible layer .

The storing patterns for QBAAM are then defined. The patterns of the network is

a combination of quaternionic values (for visible layer) and real values (for invisible layer), thus the pairs of storing patterns are necessary. Suppose that the patterns for visible layers are given by $\xi_v^1, \xi_v^2, \ldots, \xi_v^P$. For each of these patterns, the patterns for invisible layers are also given by $\xi_i^1, \xi_i^2, \ldots, \xi_i^P$. The storing patterns for QBAAM are $(\xi_v^1, \xi_i^1), (\xi_v^2, \xi_i^2), \cdots, (\xi_v^P, \xi_i^P)$. The storing pattern matrices are

$$\boldsymbol{X} = \left(\xi_v^1, \xi_v^2, \cdots, \xi_v^P\right), \tag{3.16}$$

$$\boldsymbol{Y} = \left(\xi_i^1, \xi_i^2, \cdots, \xi_i^P\right). \tag{3.17}$$

The bidirectional connection weights are constructed from the matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$. The most straightforward algorithm for the construction is known as the Hebbian learning rule:

$$\boldsymbol{W}_{iv} = \boldsymbol{Y}\boldsymbol{X}^*, \tag{3.18}$$

$$\boldsymbol{W}_{vi} = \boldsymbol{X}\boldsymbol{Y}^*, \tag{3.19}$$

where $\boldsymbol{W}_{iv}$ is a connection matrix from the visible to invisible layers and $\boldsymbol{W}_{vi}$ is a connection matrix from the invisible to visible layers, and these matrices are Hermitian, i.e., $\boldsymbol{W}_{iv} = \boldsymbol{W}_{iv}{}^*$ holds. This learning scheme makes the embedded patterns in the network unstable and leads to extremely low storage capacity if the patterns are not orthogonal to each other. In order to improve the storage capacity, generalized inverse matrix learning scheme was proposed in [25], given as

$$\boldsymbol{W}_{iv} = \boldsymbol{Y}\left(\boldsymbol{X}^*\boldsymbol{X}\right)^{-1}\boldsymbol{X}^*, \tag{3.20}$$

$$\boldsymbol{W}_{vi} = \boldsymbol{X}\left(\boldsymbol{Y}^*\boldsymbol{Y}\right)^{-1}\boldsymbol{Y}^*. \tag{3.21}$$

This scheme first orthogonalized the embedded patterns, after which the orthogonalized patterns are embedded by Hebbian rule. Though $\boldsymbol{W}_{iv} = \boldsymbol{W}_{iv}^*$ does not hold, this scheme effectively works.

Retrieving the embedded patterns from the input patterns conducts in a ping-pong manner between neurons in the visible layer and neurons in the invisible layer. Let $\boldsymbol{x}(t) = (x_1(t), x_2(t), \ldots, x_N(t))^{\mathrm{T}}$ be states of the neurons in the visible layer, and let

$\boldsymbol{y}(t) = (y_1(t), y_2(t), \ldots, y_M(t))^{\mathrm{T}}$ be states of the neurons in the invisible layer at the time step t. The states of each layer are updated as follows:

$$\boldsymbol{y}(t) = \mathrm{sign}\left(\boldsymbol{W}_{iv}\boldsymbol{x}(t)\right), \tag{3.22}$$

$$\boldsymbol{x}(t+1) = \mathrm{qsign}\left(\boldsymbol{W}_{vi}\boldsymbol{y}(t)\right), \tag{3.23}$$

where siqn$(\cdot)$ and qsign$(\cdot)$ are activation functions for neurons in the invisible layer and visible layer, respectively, and these activation functions are applied to each neurons. sign is defined as

$$\mathrm{sign}(u) = \begin{cases} 1 & \mathrm{Re}(u) \geq 0 \\ -1 & \mathrm{Re}(u) < 0 \end{cases}, \tag{3.24}$$

and qsign$(\cdot)$ is the same definition as Eq. (3.3). At the initial states, it is necessary to set an input pattern on the neurons in the visible layers. The initial configuration in the visible layer and connection weights makes the neurons' states in the invisible layers, after which the neurons in the visible layer can be updated by the neurons' states in the invisible layer and the connection weights from the invisible to visible layers. The updating of the network continues until no changes being detected at the neurons in the visible layer. Finally we can obtain the pattern ξ_v in the visible layer which corresponds to the input pattern to the network. These recall process is summarized in Fig. 3.4.

3.4 Quaternionic Hopfield Associative Memory with Dual Connections

This section presents quaternionic Hopfield associative memory with dual connections (QHAMDC). We can consider two types of multiplication order of operations to calculate weighted input because of non-commutative nature of quaternion algebra. All neurons are connected to each other and they have two types of connection weights as shown in Fig. 3.5. In QHAMDC, two types of connection weights are used for updating

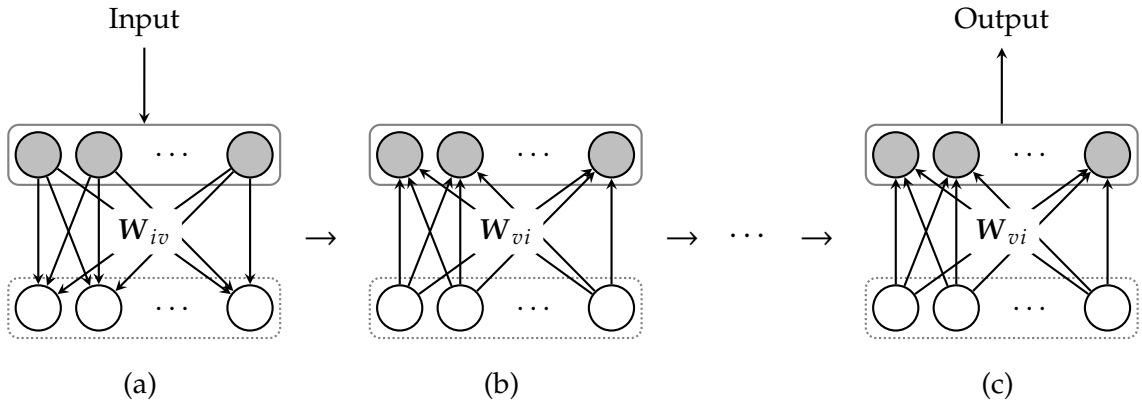


Figure 3.4: Recall process of QBAAM. (a) An input pattern is given to the visible layer and the states of the neurons in the invisible layer are updated by using the connection weights from the visible layer to the invisible layer. (b) The states of the neurons in the invisible layer are updated by using the connection weights from the invisible layer to the visible layer. (c) After iterating step (a) and step (b), the states of the neurons in the visible layer are obtained as output pattern.

the internal states of neurons:

$$h_p(t) = \sum_{q=1}^{N} \left(w_{pq}^{\mathrm{L}} u_q(t) + u_q(t) w_{pq}^{\mathrm{R}} \right), \tag{3.25}$$

where w_{pq}^{L} is a connection weight from p-th neuron to q-th neuron which is multiplied from the left and w_{pq}^{R} is one which is multiplied from the right. The connection weight matrix $\boldsymbol{W}^{\mathrm{L}}$ whose element is w_{pq}^{L} is given by Eq. (3.15). The connection weight matrix $\boldsymbol{W}^{\mathrm{R}}$ whose element is w_{pq}^{R} is given as follows:

$$\boldsymbol{W}^{\mathrm{R}} = \boldsymbol{B} - \mathrm{diag}(\boldsymbol{B}), \quad \boldsymbol{B} = \boldsymbol{Y}^{*}(\boldsymbol{Y}\boldsymbol{Y}^{*})^{-1}\boldsymbol{Y}, \quad \boldsymbol{Y} = \boldsymbol{X}^{\mathrm{T}}, \tag{3.26}$$

where $\boldsymbol{X}$ is the training pattern matrix. Thus, $\boldsymbol{W}^{\mathrm{R}}$ denotes the connection weights obtained by using the projection rule with multiplications in the reverse order. The state of a neuron in QHAMDC is updated by Eq. (3.3) in the same manner as in QHAM.

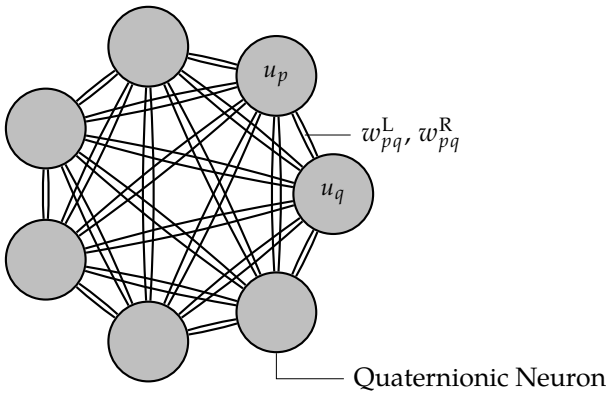


Figure 3.5: Network structure of QHAMDC.

We define the energy function of the network with dual connections as

$$E(t) = -\frac{1}{2}\sum_{p=1}^{N}\sum_{q=1}^{N}\left(u_p^*(t)w_{pq}^{\mathrm{L}}u_q(t) + u_q(t)w_{pq}^{\mathrm{R}}u_p(t)^*\right), \tag{3.27}$$

where E takes real value ($E = E^*$) when w_{pq}^{L} and w_{pq}^{R} satisfy the condition Eq. (3.12). Since $\mathrm{Re}(xy) = \mathrm{Re}(yx)$ holds for $x, y \in \mathbb{H}$, the energy function can be represented as

$$E(t) = -\frac{1}{2}\,\mathrm{Re}\sum_{p=1}^{N}\sum_{q=1}^{N}u_p^*(t)\left(w_{pq}^{\mathrm{L}}u_q(t) + u_q(t)w_{pq}^{\mathrm{R}}\right). \tag{3.28}$$

Suppose that the state of the r-th neuron is updated at time $t+1$ according to the Eq. (3.25) and Eq. (3.3). The energy gap ΔE between the $E(t+1)$ and $E(t)$ finally becomes as follows:

$$\begin{aligned}\Delta E = &-\mathrm{Re}\sum_{r=1}^{N}\sum_{q=1}^{N}u_r^*(t+1)\left(w_{rq}^{\mathrm{L}}u_q(t) + u_q(t)w_{rq}^{\mathrm{R}}\right)\\ &+\mathrm{Re}\sum_{r=1}^{N}\sum_{q=1}^{N}u_r^*(t)\left(w_{rq}^{\mathrm{L}}u_q(t) + u_q(t)w_{rq}^{\mathrm{R}}\right).\end{aligned} \tag{3.29}$$

ΔE becomes non-positive because $u_r^*(t)$ maximize $\mathrm{Re}(u_r^*(t)h_r(t))$ according to the activation function. Therefore, the energy decreases monotonically.

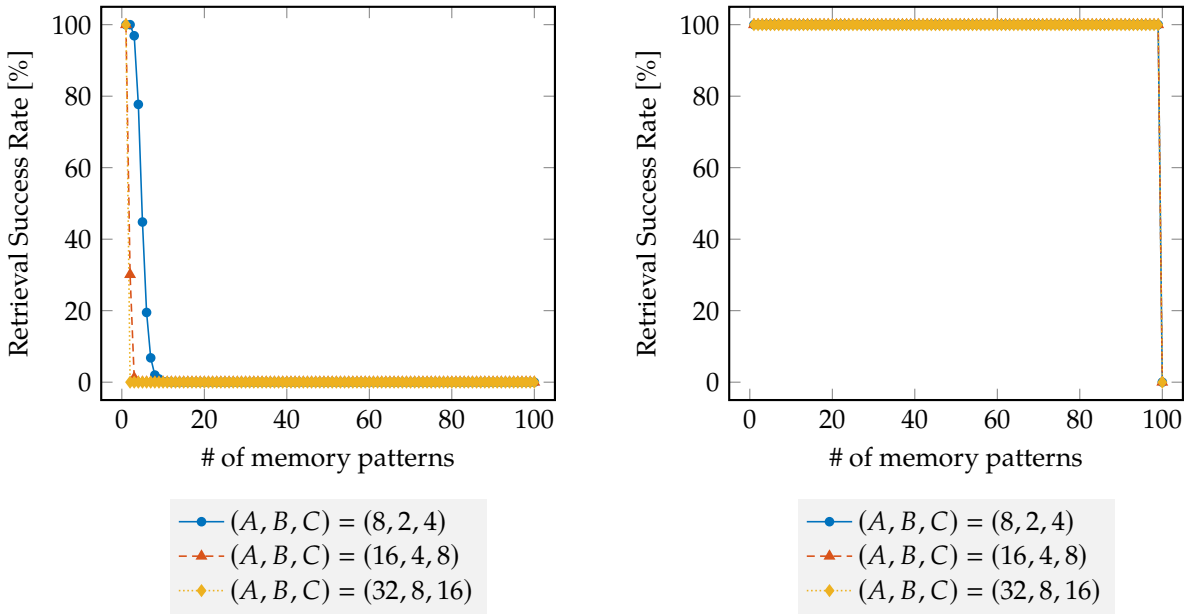
3.5 Experiments

3.5.1 Storage Capacity

We first evaluate the storage capacity of QHAM by using the Hebbian learning rule and the projection learning rule. In this experiment, the patterns with randomly generated values are used as memory patterns. The size of the patterns, which is the number of neurons in the network, was set to $N = 100$, and the quantization level was set as follows: $(A, B, C) = (8, 2, 4), (16, 4, 8), (32, 8, 16)$. In these conditions, the quantization units φ_0, ψ_0, and θ_0 are the same size. The number of the memory patterns, denoted by P, varies such that $P = 1, 2, \ldots, 100$.

The stability of the patterns are investigated by the following procedure. First, for given A, B, C, and P, memory patterns are generated and stored into the network. Next, each of the memory patterns is set to the network as its initial states, then the states for all neurons are updated. If the network state does not change, this stored pattern can be regarded as stable.

Figure 3.6(a) shows P dependency of the retrieval success rates against various quantization level. The retrieval success rates are calculated from 1000 trials. From Fig. 3.6(a), we find that the memory patterns are hardly embedded to the network by using the Hebbian learning rule. The memory patterns tend to be more unstable with increases of the quantization level and the number of stored patterns. If the quantization level (A, B, C) is set to $(32, 8, 16)$, only one pattern is stable in the network as shown in Fig. 3.6(a). That is, two or more memory patterns cannot to be stable in the network by using the Hebbian rule for large quantization level. In contrast, the projection rule can store up to 99 patterns in the network regardless of the quantization levels as shown in Fig. 3.6(b). In this figure, the retrieval success rate is 0% in the case of $P = 100$. It is due to that the self-connection weights, $w_{pp}s$, are set to 0. If these weights are set to positive values, the retrieval success rate becomes 100%. From these results, all the stored patterns are local minima in the network by using the projection



(a) Hebbian learning rule

(b) Projection learning rule

Figure 3.6: Stability of the stored memory patterns in QHAM ($N = 100$).

learning rule. Therefore the memory patterns stored by using the projection rule have higher stability than those by using the Hebbian rule in QHAM.

3.5.2 Noise Robustness of Recall

We evaluate the recall performance by retrieving patterns from noisy patterns. The experiments for QHAMs, QBAAMs, and QHAMDCs are conducted with the following parameters:

1. The number of neurons in the network (N) is set to 100.
2. The number of patterns to be embedded to the network (P) is set to 10, 30, and 50.
3. The quantization levels for quaternionic neurons (A, B, C) are set to $(8, 2, 4)$, $(16, 4, 8)$, and $(32, 8, 16)$.
4. The noise level (r) changes to 0.00, 0.05, $\cdots$, and 0.80.
5. The update of neurons in the network is repeated for 1000 iterations.
6. For each parameter, 1000 experiments are conducted with randomly generated patterns being embedded.

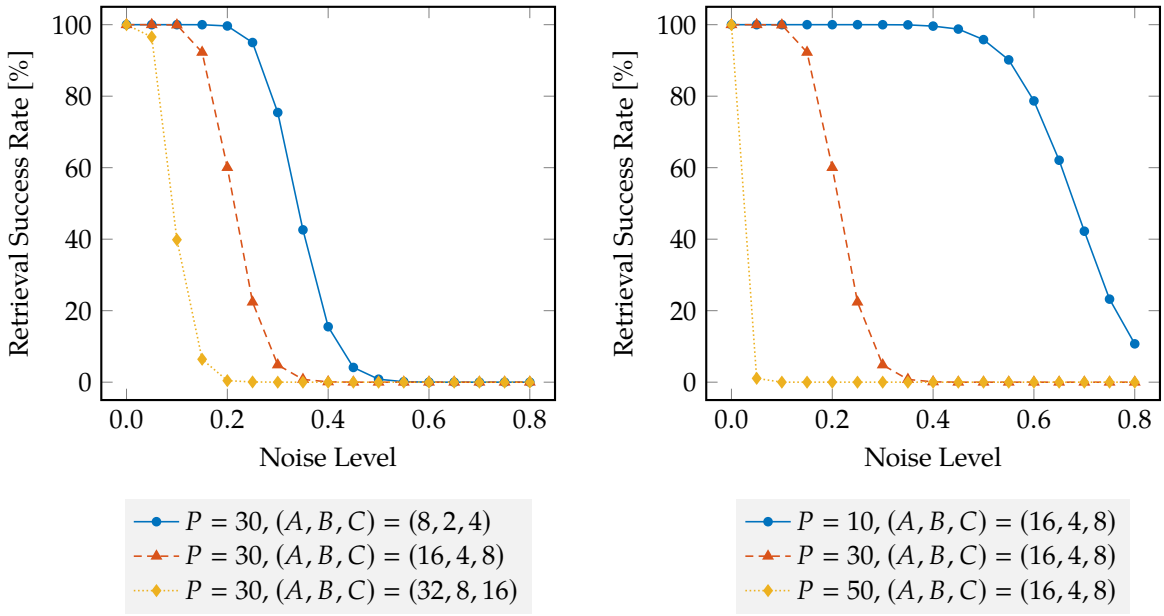


Figure 3.7: Noise robustness of recall performance for QHAM ($N = 100$)

The memory patterns are randomly generated and they are embedded to the each associative memory by using projection learning rule. The noisy input pattern is generated from one of the memory patterns with each pixel value being changed to random value by a specific ratio r (noise level). The retrieval is regarded as successful if the correct pattern corresponding to the input pattern can be retrieved from the noisy input pattern.

First, we examine the performance of QHAMs. Figure 3.7 shows the retrieval success rates against the noise level. The number of memory patterns was fixed to $P = 30$ and the quantization level was set to $(A, B, C) = (8, 2, 4), (16, 4, 8), (32, 8, 16)$ in Fig. 3.7 on the left. The number of memory patterns was set to $P = 10, 30, 50$ and the quantization level was fixed to $(A, B, C) = (16, 4, 8)$ in Fig. 3.7 on the right. It is shown that the retrieval success rates for QHAMs get worse when the number of embedded patterns increases or the quantization levels increases. The noise robustness of QHAM highly depends on the quantization levels. Thus, the quantization level is particularly sensitive parameters for the performance of QHAMs.

Figure 3.8 shows the retrieval success rates against the noise rate on QBAAMs. In this

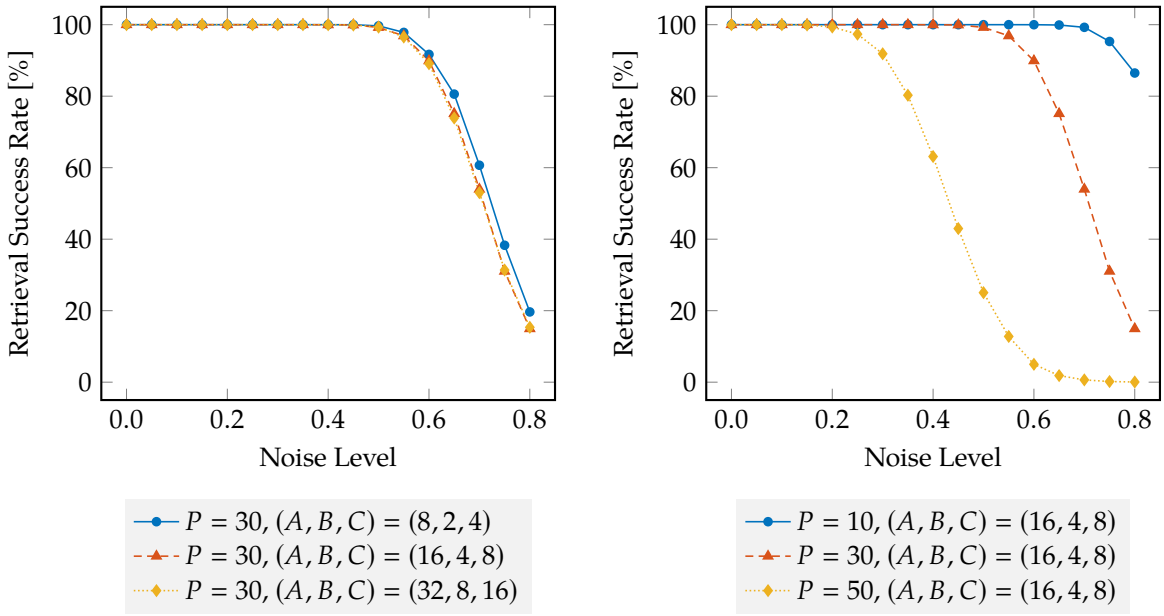


Figure 3.8: Noise robustness of recall performance for QBAAM ($N = 100, M = 100$)

experiment, the number of neurons in the invisible layer is set to $M = 100$. From these results, we find that the retrieval success rates are maintained when the quantization levels are increased. Thus, the noise robustness of QBAAM does not depend on the quantization levels. In QBAAMs, the noise robustness can be controlled by changing the number of neurons in the invisible layer. Figure 3.9 shows the dependency of the success rate on the number of neurons in the invisible layer. The parameters for the network were set to $(A, B, C) = (16, 4, 8)$ and $P = 30$. The number of neurons in the invisible layer was changed from $M = 20$ to $M = 180$. This result shows that neurons in the invisible layer assist to retrieve the correct pattern.

Finally, the recall perforamnce for QHAMDCs is shown in Figure 3.10. From these results, we find that the retrieval success rates are maintained when the quantization levels are increased similar to that of QBAAM. Also, the noise robustness of QHAMDC does not depend on the quantization levels.

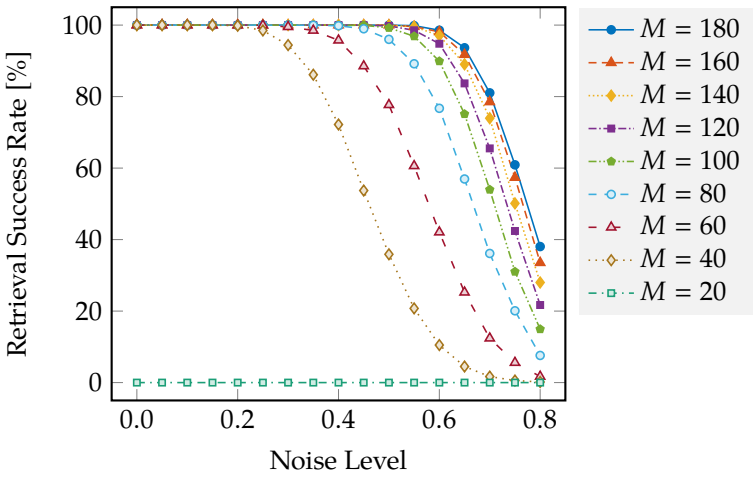


Figure 3.9: Dependency of recall performance on the number of neurons in the invisible layer for QBAAM ($N = 100, P = 30, (A, B, C) = (16, 4, 8)$)

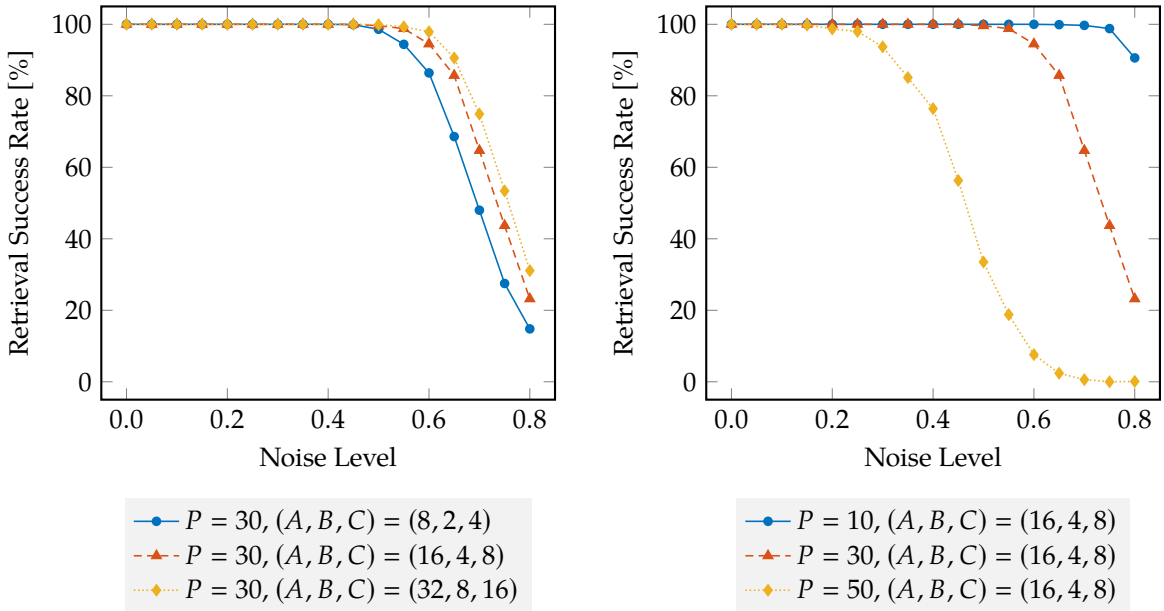


Figure 3.10: Noise robustness of recall performance for QHAMDC ($N = 100$)

3.5.3 Image Retrieval Task

We have explored the noise robustness of three types of quaternionic associative memories by using random patterns in the previous experiments. In this section, we investigate the performances of by storing and retrieving natural images, which have more intensity resolutions in each pixel, from the viewpoint of practical applications such as color image database.

Figure 3.11 shows memory patterns to be embedded to the network for this experiments. These images are 200 images included in CIFAR-10 dataset [26]. Each image consists of $32 \times 32 = 1024$ pixels and each pixel value is represented by 24-bit for RGB color. The three channels, which are red, green, and blue, are assigned to φ, θ, and ψ of a quaternion in polar representation. Thus, the quantization level is set to $(A, B, C) = (256, 256, 256)$. For example, a tuple of RGB pixel value (a, b, c) is represented as a quaternion $q_a^{(\varphi)} q_b^{(\psi)} q_c^{(\theta)}$ based on Eq. 3.7, Eq. 3.10, and Eq. 3.11. The number of neurons in the network is the same as the number of pixels of the images, i.e., $N = 1024$. In this experiment the number of neurons in the invisible layer for the QBAAM was set to $M = 1024$, thus the degree of freedom of the connection weights in the QBAAM and QHAMDC is equal. The memory patterns are embedded to the networks with the projection learning rule. Then, noisy patterns as shown in Fig. 3.12 are set as the initial configuration of the network and the output patterns are obtained after the 1000 iterations of the update process.

Figures 3.13, 3.14, and 3.15 show recalled output patterns for QHAM, QBAAM, and QHAMDC, respectively. From these figures, it is found that the QHAM cannot retrieve the stored pattern in all patterns. In contrast, the recalled patterns in the QBAAM and QHAMDC are closer to the truth compared to those in the QHAM. These are confirmed by the peak signal-to-noise ratio (PSNR) shown in Table 3.1, which is defined as,

$$\mathrm{PSNR} = 20 \log_{10} (255) - 10 \log_{10} (\mathrm{MSE}) . \tag{3.30}$$

where, MSE is mean squared error of the original image and the recalled one. The averaged PSNR of the patterns retrieved from the QHAM and the corresponding original

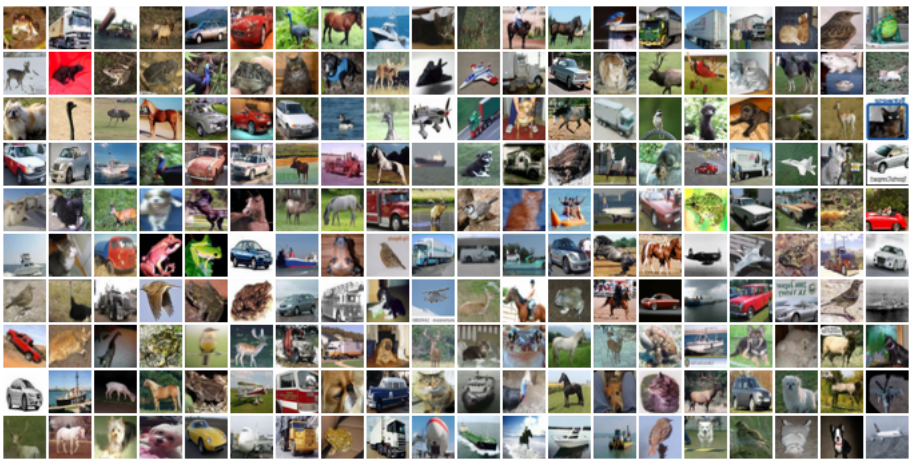


Figure 3.11: Memory patterns for image retrieval task ($P = 200$).

Figure 3.12: Input patterns for associative memories. 50 % of all pixels in each image are altered by random values.

patterns is lower than that of the original patterns and the noisy patterns. This result is caused by the spurious patterns due to the rotation invariance of memory patterns in QHAMs. These spurious patterns are suppressed to be retrieved due to the real-valued neurons in QBAAMs or the non-commutative property of quaternionic connection weights in QHAMDCs, so that he higher PSNRs were achieved in the QBAAM and QHAMDC.

Figure 3.13: Output patterns of QHAM ($N = 1024$).

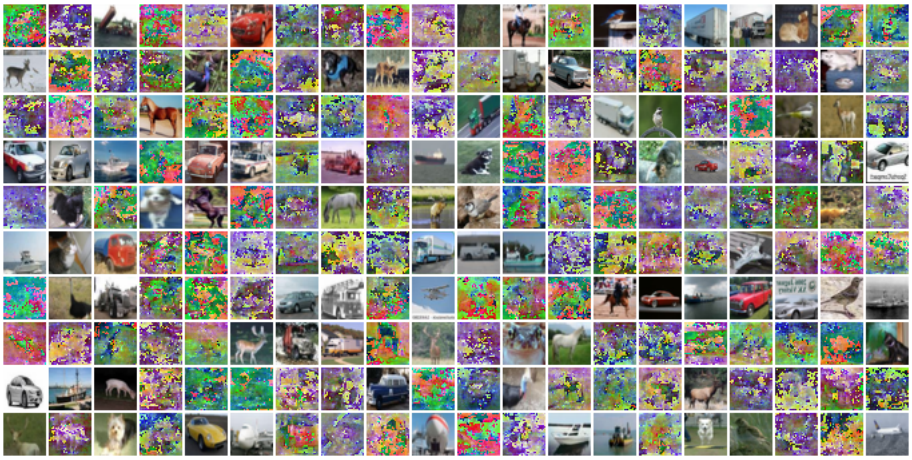


Figure 3.14: Output patterns of QBAAM ($N = 1024, M = 1024$).

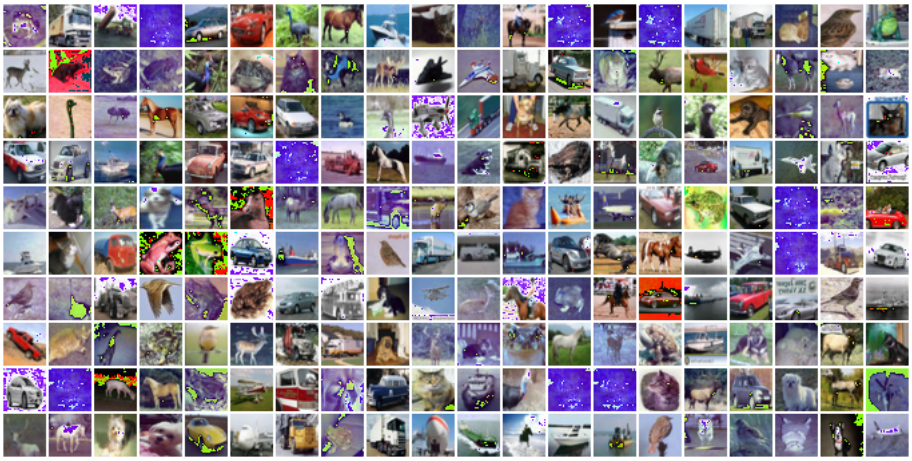


Figure 3.15: Output patterns of QHAMDC ($N = 1024$).

Table 3.1: Averaged PSNR for all recalled images.

Noisy Input	11.42 [dB]
QHAM	10.77 [dB]
QBAAM ($M = 1024$)	12.81 [dB]
QHAMDC	16.86 [dB]

3.6 Conclusion

In this chapter, we investigate the performance of quaternionic associative memory models. First, the storage capacity of QHAM by using Hebbian learning rule and projection learning rule is examined. From the experimental results, the Hebbian rule hardly stores the memory patterns with increasing the quantization levels of quaternionic neurons. In contrast, the projection rule can stabilize all the memory patterns into the network regardless of quantization levels.

The noise robustness of the retrieval patterns is also investigated and it is found that the performance from noisy input depends on the quantization levels of the quaternionic neuron and the number of stored memory patterns. The quantization level is particularly sensitive parameter for the recall performance of QHAMs.

In order to obtain better performances in retreiving patterns, we propose quaternioni bipartite auto-associative memory (QBAAM). The recall performance of QHAMs suffer from the spurious patterns caused by the rotation invariance of their representation of neuron states. The proposed model has two-layered network structure where one layer is the visible layer and contains quaternionic neurons and the neurons in the other layer, called invisible layer, are real-valued neurons. A combination of real-valued neurons and quaternionic neurons suppresses these spurious patterns and leads to higher noise robustness in the networks.

we also propose an another type of quaternionic associative memory model, quaternionic Hopfield associative memory with dual connections (QHAMDC). Two types of

connection weights are used in QHAMDC: one are connection weights multiplied from the left and the other are connection weights multiplied from the right. Noise robustness of the conventional quaternionic Hopfield associative memory (QHAM) is deteriorated by the spurious patterns caused by rotational invariance of training memory patterns. In QHAMDC, rotated patterns, which is one of the typical spurious patterns, can be reduced by a combination of two types of connection weights utilizing the non-commutative nature of quaternions. It is shown that the noise robustness for retrieving patterns in QHAMDC is superior to those in QHAM from experimental results. More detailed analysis on spurious patterns in QHAM, QBAAM, and QHAMDC remains for our future work.

Chapter 4

Pseudo-Orthogonalization of Memory Patterns for Quaternionic Hopfield Neural Network

Hebbian learning rule is well known as a memory storing method for associative memories. This method is simple and fast, however, its performance gets decreased when memory patterns are not orthogonal to each other. Pseudo-orthogonalization is a one solution for embedding these non-orthogonal memory patterns into associative memories. By a combination of the pseudo-orthogonalization and Hebbian learning rule, storage capacity of associative memory concerning non-orthogonal patterns is improved without high computational cost. The memory patterns can also be retrieved based on a simulated annealing method from an external stimulus pattern. By utilizing quaternions, we can extend the pseudo-orthogonalization scheme for quaternionic Hopfield neural networks. In this study, the extended pseudo-orthogonalization scheme for associative memories based on quaternions. We show that the proposed scheme has stable recall performance on highly correlated memory patterns compared to the conventional real-valued scheme.

4.1 Introduction

Hebbian learning rule is a well-known method for embedding patterns onto associative memories, such as Hopfield neural networks [27, 14]. This method is simple and straightforward, however, it has a crucial issue for the embedding patterns; the patterns should be orthogonal to each other. On embedding correlated patterns by this method, the storage performance of the network is significantly decreased. Thus, many researches for storing correlated patterns direct to orthogonalization of these patterns, such as pseudo-inverse matrix method [24] and iterative learning method [28]. Though these methods enable all the correlated patterns to be stable local minima in the network, their computational costs grow with respect to the network size and the number of patterns to be embedded.

A novel method has been proposed that enables correlated patterns onto associative memories with low computational cost [29]. The method called pseudo-orthogonalization first prepares a random pattern (mask pattern) of which length is the same as that of a memory pattern, and element-wise exclusive not-or (XNOR) operation is applied between the random pattern and the memory pattern. The pattern to be embedded in the network is a concatenation of XNORed pattern (masked pattern) and the corresponding random pattern. These pseudo-orthogonalized patterns have low correlation compared to that of the original memory patterns, thus, they can be embedded by using Hebbian learning rule without a degradation of storage capacity. The length of these patterns become double, however, the embedding process is simple and requires rather low computational cost.

The complex-valued or quaternionic extensions would be suitable for the pseudo-orthogonalization scheme; the pair of a pattern can be naturally embedded by utilizing imaginary part(s). In this study, we extend the pseudo-orthogonalization scheme by using quaternions. We investigate the performance of the quaternionic extended method through storing binary memory patterns to quaternionic Hopfield networks [30], and the performance is also examined from the view point of correlations in memory pat-

terns and the loading rate (the ratio of the number of memory patterns to the number of units in the network). We show that the extended method has stable recall performance on highly correlated memory patterns compared to the conventional real-valued method does.

This chapter is organized as follows. In Section 4.2, we summarize real-valued and quaternionic Hopfield neural networks. Quaternionic extension of pseudo orthogonalization scheme is described in Section 4.3. Several experimental results for evaluating the proposed scheme are given in Section 4.5. In Section 4.7, we discuss the superior performances for quaternionic pseudo-orthogonalization. We finish with conclusions in Section 4.8.

4.2 Preliminaries

In this section, we explain Hebbian learning rule and the network dynamics for real-valued and quaternionic Hopfield neural networks.

4.2.1 Real-Valued Hopfield Neural Network

Let $(\xi_1^\mu, \ldots, \xi_N^\mu)$, $\xi_m^\mu \in \{+1, -1\}$ be the μ-th learning pattern. Hebbian learning rule for real-valued Hopfield neural network (RHNN) is represented as

$$w_{mn} = \frac{1}{N} \sum_{\mu=1}^{P} \xi_m^\mu \xi_n^\mu, \tag{4.1}$$

where w_{mn} is a synaptic weight between m-th and n-th neurons, which satisfies the conditions $w_{mm} = 0$ and $w_{mn} = w_{nm}$ for all m and n, and P is the number of the learning patterns. The dynamics of the network is given as,

$$x_m(t+1) = \operatorname{sgn}\left(\sum_{n=1}^{N} w_{mn} x_n(t) \right), \tag{4.2}$$

where $x_m(t) \in \{+1, -1\}$ denotes the output of the m-th neuron at the time step t and N is the total number of neurons in the network. The function $\operatorname{sgn}(\cdot)$ is an activation function which is defined by $\operatorname{sgn}(u) = 1$ when $u \geq 0$, and $\operatorname{sgn}(u) = -1$ when $u < 0$.

4.2.2 Quaternionic Hopfield Neural Network

In quaternionic Hopfield neural network (QHNN), all neuronal parameters in the network are encoded by quaternions [30, 31]. Let the m-th element of a μ-th quaternionic learning pattern be $\xi_m^\mu = \xi_m^{\mu\,(e)} + \xi_m^{\mu\,(i)}\boldsymbol{i} + \xi_m^{\mu\,(j)}\boldsymbol{j} + \xi_m^{\mu\,(k)}\boldsymbol{k}$ where $\xi_m^{\mu\,(e)}, \xi_m^{\mu\,(i)}, \xi_m^{\mu\,(j)}, \xi_m^{\mu\,(k)} \in \{+1, -1\}$. Hebbian learning rule for QHNN is represented as

$$w_{mn} = \frac{1}{4N}\sum_{\mu=1}^{P} \xi_m^\mu \xi_n^{\mu *}, \tag{4.3}$$

where synaptic weights satisfy the conditions $w_{mm} \geq 0$ and $w_{mn} = w_{nm}^*$. The dynamics of the network is given as follows:

$$x_m(t+1) = \mathrm{qsgn}\Big(\sum_{n=1}^{N} w_{mn} x_n(t)\Big). \tag{4.4}$$

The function qsgn$(\cdot)$ is an activation function for quaternionic neurons which is defined by

$$\mathrm{qsgn}(s) = \mathrm{sgn}(s^{(e)}) + \mathrm{sgn}(s^{(i)})\boldsymbol{i} + \mathrm{sgn}(s^{(j)})\boldsymbol{j} + \mathrm{sgn}(s^{(k)})\boldsymbol{k}. \tag{4.5}$$

4.3 Pseudo-Orthogonalization based on Quaternions

The purpose of the pseudo-orthogonalization is to randomize memory patterns so that they can be stored by Hebbian learning. We present a pseudo-orthogonalization based on quaternions in this section.

First, let us recapitulate the real-valued pseudo-orthogonalization [29]. The memory patterns are masked by using random mask patterns as shown in Fig. 4.1. Fig. 4.2 shows the generation method for the masked patterns. The masked patterns are obtained by element-wise multiplication of the memory patterns and random patterns. Thus, the original patterns can be reconstructed from the masked patterns and the random patterns as shown in Fig. 4.3. The pseudo-orthogonalized pattern is obtained as the concatenation of these random mask pattern and masked pattern.

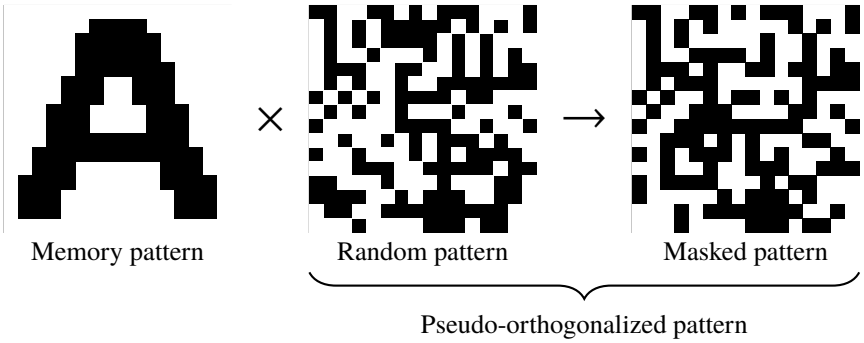


Figure 4.1: Schematic of pseudo-orthogonalization.

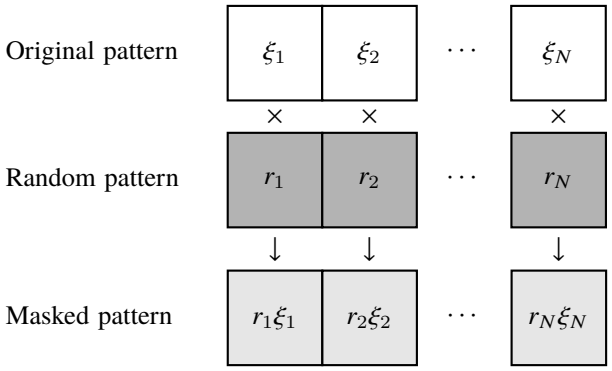


Figure 4.2: Generation of masked patterns.

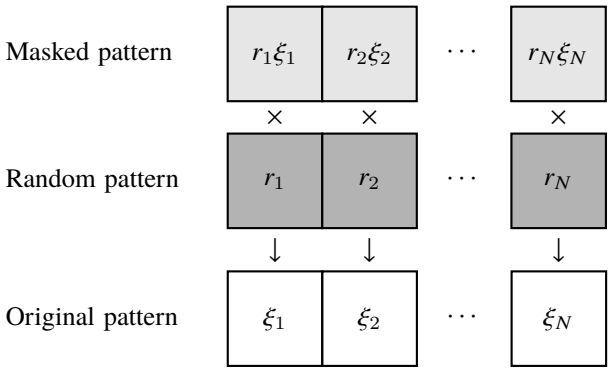


Figure 4.3: Reconstruction of original patterns.

Let $(\xi_1, \ldots, \xi_N)$ where $\xi_m \in \{+1, -1\}$ be the original memory pattern and $(r_1, \ldots, r_N)$ where $r_m \in \{+1, -1\}$) be a random pattern corresponding to the original pattern. The m-th element of the real-valued pseudo-orthogonalized pattern is generated by

$$\eta_m^{\mathrm{r}} = \begin{cases} r_n, & m = 2n-1 \\ r_n \xi_n, & m = 2n \end{cases}. \tag{4.6}$$

This is the concatenation of the random pattern and the masked pattern between the original memory pattern and the random pattern (see Fig. 4.4(a)). Thus, The element η_m takes either +1 or −1 and the length of the pseudo-orthogonalized pattern, denoted as N', becomes twice as the original one, i.e. $N' = 2N$.

Next, we show extended pseudo-orthogonalization by using quaternions. The m-th element of the quaternionic pseudo-orthogonalized pattern is defined by

$$\eta_m^{\mathrm{q}} = r_{2m-1} + r_{2m-1}\xi_{2m-1}\boldsymbol{i} + r_{2m}\boldsymbol{j} + r_{2m}\xi_{2m}\boldsymbol{k}. \tag{4.7}$$

That is, odd numbered elements in random patterns are assigned to the real part and the rest is assigned to the imaginary part $\boldsymbol{j}$. Also, odd numbered elements in masked patterns are assigned to the imaginary part $\boldsymbol{i}$ and the rest is assigned to the imaginary part $\boldsymbol{k}$. Therefore, the original patterns can be reconstructed by

$$\xi_m = \begin{cases} \eta_n^{\mathrm{q}(e)}\, \eta_n^{\mathrm{q}(i)}, & m = 2n-1 \\ \eta_n^{\mathrm{q}(j)}\, \eta_n^{\mathrm{q}(k)}, & m = 2n \end{cases}. \tag{4.8}$$

$N' = N/2$ is obtained in the quaternionic pseudo-orthogonalization (see Fig. 4.4(b)).

4.4 Retrieval Dynamics for Quaternionic Pseudo Orthogonalization

In this section, we describe the retrieval dynamics for pseudo-orthogonalization scheme.

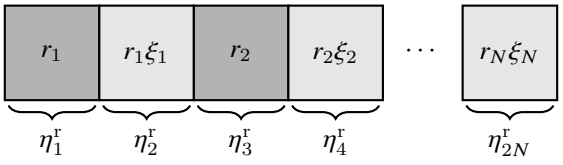
In Hopfield network, a pattern to be a clue for recall is set as the initial state of the network, and then the memory pattern is retrieved after iterations of updating the states

of neurons. However, the information of random mask pattern which is used in the pseudo-orthogonalization is unknown in the retrieval process, so that the initial state of the network cannot be determined in pseudo-Orthogonalization scheme. Therefore, the network dynamics to be extended for retrieving memory patterns without random mask patterns. By using a simulated annealing method, the recall dynamics for real-valued psuedo-orthogonalizaion is defined as follows [29]:

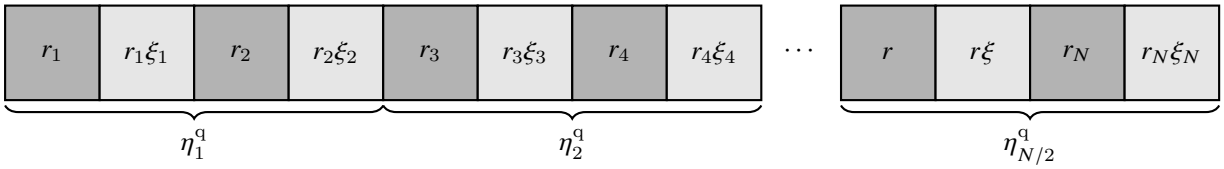
$$h_i(t) = \begin{cases} \sum_{n=1}^{N} w_{ij} x_j(t) + s z_i x_{2k}, & i = 2k-1 \\ \sum_{n=1}^{N} w_{ij} x_j(t) + s z_i x_{2k-1}, & i = 2k \end{cases} \tag{4.9}$$

$$\text{Prob}(x_m^{(*)} = 1) = \frac{1}{1 + \exp(-\beta(t) h_m(t))}. \tag{4.10}$$

Similarly, we can extend the dynamics for quaternionic pseudo-orthogonalization, and



(a) Real-valued pseudo-orthogonalization



(b) Quaternionic pseudo-orthogonalization

Figure 4.4: Pseudo-orthogonalized patterns generated from random patterns and masked patterns utilizing (a) real numbers and (b) quaternions.

it is formulated as following equations:

$$h_m(t)=\sum_{n=1}^{N} w_{mn}x_n(t) + sz_{2m-1}\left(x_m^{(i)}(t)+x_m^{(e)}(t)\boldsymbol{i}\right) + sz_{2m}\left(x_m^{(k)}(t)\boldsymbol{j}+x_m^{(j)}(t)\boldsymbol{k}\right), \tag{4.11}$$

$$\mathrm{Prob}(x_m^{(*)}=1) = \frac{1}{1+\exp(-\beta(t)h_m^{(*)}(t))}, \quad (*)\in\{(e),(i),(j),(k)\}. \tag{4.12}$$

where $z_m \in \{+1,-1\}$ denotes the m-th element of an external stimulus which is to be cue signal pattern, s is the strength of the external stimulus, and $\beta(t+1)=\gamma\beta(t)$ is the inverse temperature parameter that increases with time step t. γ with $(\gamma > 1)$ is the increase rate for β. The states of the neurons $x_m(t)$ are initialized randomly at $t=0$ and they evolve stochastically by using Eq. (4.12) Here, the real part and the imaginary part of the internal state $h_m(t)$ are separately updated.

4.5 Experiments

4.5.1 Recalling patterns by using extended dynamics

First, We show recall capabilities of the networks from an external stimulus pattern by using the extended dynamics defined by Eq. 4.12. In this experiment, 6 kanji patterns are used as memory patterns as shown in Fig. 4.5. Each pattern is constituted by 42×42 pixels. These images are strongly correlated, because they have a same local pattern on their left hand. Thus, these patterns cannot be stabilize in Hopfield networks by using Hebbian learning rule.

Figure 4.6 shows the retrieval result by using real-valued and quaternionic pseudo-orthogonalization scheme. Fig. 4.6(a) shows the evolutions of overlaps between ground

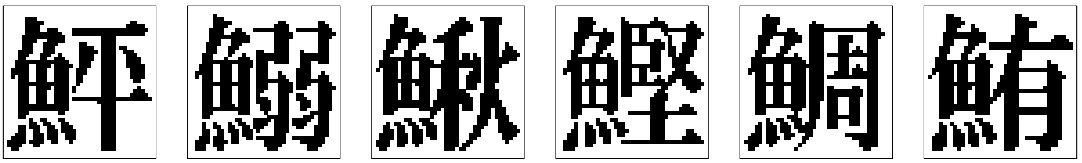


Figure 4.5: Kanji images ($N = 42\times42 = 1764$, $P = 6$)

truth pattern (鮮) and recalled pattern. The overlap is defined by the average of correlations among the patterns stored in the network. We define the overlap o_{r} for real-valued patterns and o_{q} for quaternionic patterns are defined as

$$o_{\mathrm{r}} = \frac{1}{N}\sum_{m=1}^{N} \xi_m^{\mu} x_m(t), \tag{4.13}$$

$$o_{\mathrm{q}} = \frac{1}{4N}\sum_{m=1}^{N} \left(\xi_m^{\mu\,(e)} x_m(t)^{(e)} + \xi_m^{\mu\,(i)} x_m(t)^{(i)} + \xi_m^{\mu\,(j)} x_m(t)^{(j)} + \xi_m^{\mu\,(k)} x_m(t)^{(k)} \right). \tag{4.14}$$

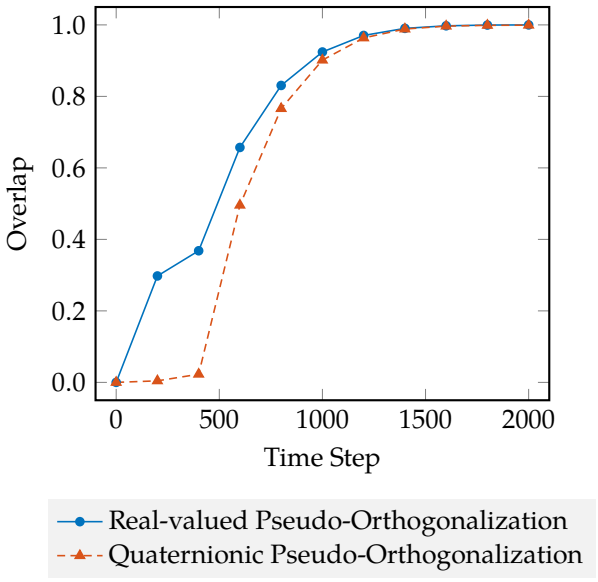
In this results, the parameters for updating state of the neurons in the network were set to $\beta(0) = 1.0$, $\gamma = 1.001$, $s = 1.0$, and the pattern shown in Fig 4.6(b) is used as the external stimulus. At the final step($t = 2000$), the overlaps for each scheme become nearly 1.0, so that the original memory pattern are retrieved successfully. This can be confirmed from Figs. 4.6(c) and 4.6(d). These figures are reconstructed images from the pseudo-orthogonalized patterns recalled in the network for each time step.

Figure 4.7 shows the retrieval result when a noisy pattern is used as external stimulus. In this results, the strength for external stimulus is set to $s = 0.5$ to avoid attracting the neuron state to the noise component excessively. The noisy external stimulus is shown in Fig. 4.6(b). This pattern is generated by inverting the pixel value of 30% of the original pattern. We find that even if the noisy patterns are given to the network as external stimulus, the pattern corresponding to the stimulus is successfully recalled as shown the overlap and reconstructed images. Therefore, the quaternionic extended recall dynamics functions correctly.

4.6 Retrieval performance

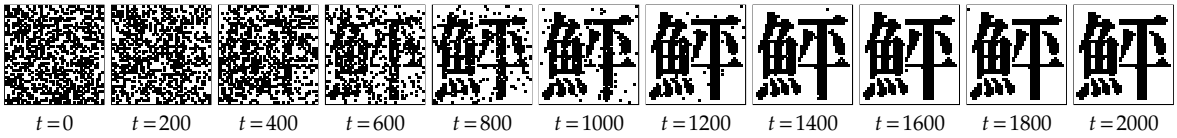
Next, we investigate the retrieval performance by using the proposed scheme compared to the conventional real-valued scheme. For this experiment, random patterns are used as the original memory pattern which are obtained by using the following probability:

$$\mathrm{Prob}(\xi_m = \pm 1) = (1 \pm \sqrt{b}\,\zeta_m)/2, \tag{4.15}$$

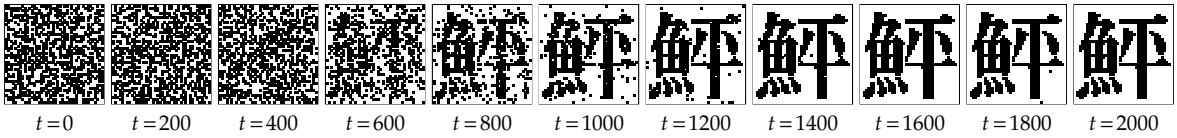


(a) Overlap

(b) External stimulus (Input pattern)

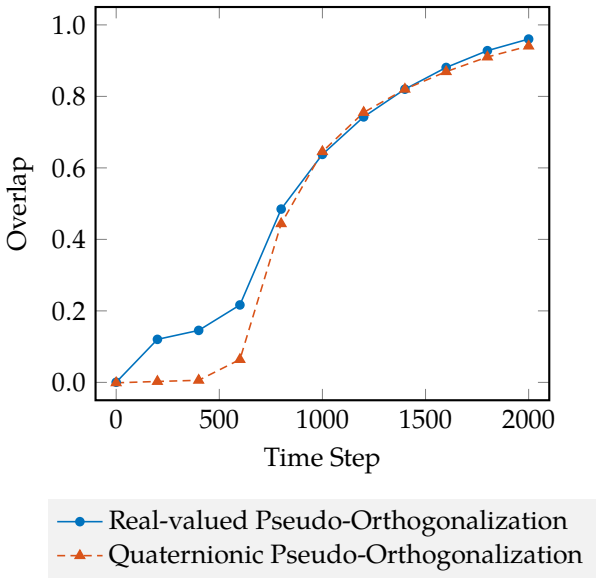


(c) Retrieved patterns by real-valued pseudo-orthogonalization scheme



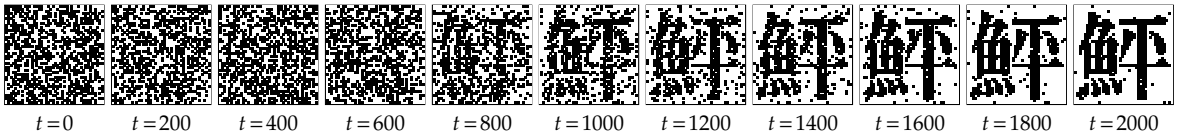
(d) Retrieved patterns by quaternionic pseudo-orthogonalization

Figure 4.6: Time Evolutions of recall process ($\beta(0) = 1.0$, $\gamma = 1.001$, $s = 1.0$).



(a) Overlap

(b) External stimulus (30% of original pattern are inverted)



(c) Retrieved patterns by real-valued pseudo-orthogonalization scheme

(d) Retrieved patterns by quaternionic pseudo-orthogonalization scheme

Figure 4.7: Time Evolutions of recall process ($\beta(0) = 1.0$, $\gamma = 1.001$, $s = 0.5$).

where ζ_m is the m-th element in a random pattern generated according to the uniform probability:

$$\text{Prob}(\zeta_m = \pm 1) = 1/2. \tag{4.16}$$

b is a correlation parameter for the random patterns which satisfies $\text{E}[\text{Corr}(\xi_m, \zeta_m)] = b$.

First, we investigated how the correlation in the original memory patterns affects the retrieval performance for pseudo-orthogonalization scheme. Figure 4.8 shows the retrieval success rates with changing the correlation parameter b of original memory patterns. In this experiments, the length of the original memory patterns was set to 1000, so that the number of neurons in RHNNs and QHNNs were set to 2000, and 500, respectively. The loading rate, which is the ratio of the number of the stored patterns P and the number of neurons N, was fixed to 0.13, and the correlation parameter was set to 0.1 from 0.0 to 0.5 with a step of 0.05. The parameters for the recall process were set as $s = 0.9$, $\gamma = 1.002$, $\beta(0) = 1.0$. We obtained the retrieval pattern after 1000 iterations of updates and 100 trials were conducted. For each experiment, an initial configuration of the network is determined randomly, and one of the original memory patterns is used as the external stimulus. The recall was considered to be successful when the overlap between the retrieved pattern and its true pattern achieved 0.95. From the figure, we find that the retrieval success rates are decreased with the increase of the correlation for the original memory patterns in all types of networks. However, the success rates in quaternionic pseudo-orthogonalization scheme are decreased slower than those in real-valued scheme.

Next, we show the dependency of the critical loading rate on the correlation in the memory patterns. The critical loading rate is defined as the loading rate when the overlaps of the retrieved pattern and its original pattern is lower than a threshold. Figure 4.9 shows the critical loading rates against the correlations. The threshold was set to 0.95 in this result. From this figure, we find that the critical loading rates in quaternionic pseudo-orthogonalizaion scheme are also decreased slower than those in real-valued scheme. The higher critical loading rate means that memory patterns can be stably embedded in and recalled from the network in the same loading rate. Therefore,

pseudo-orthogonalizaion scheme utilizing quaternions is robust to the correlation of the memory patterns compared to that of real-valued scheme.

4.7 Discussion

We discuss reasons why the retrieval performance in pseudo-orthogonalization is maintained by utilizing quaternions. First, we evaluate the dependence on the correlation of original memory patterns to examined the stability of pseudo-orthogonalized patterns in RHNNs and QHNNs.

Figure 4.10 shows the difference of the stability of pseudo-orthogonalized patterns in RHNN and QHNN. The changes of critical loading rates with increasing the correlation of the original memory patterns are shown in Fig. 4.11. The number of neurons in RHNNs and QHNNs was set to 1000. The overlaps were obtained by averaging 100 trials in 1000 updates. In each of these trials, an initial configuration are set to one of the memory patterns (pseudo-orthogonalized patterns), and the neurons are updated by using Eqs. (4.2) and (4.4). From these figures, the memory patterns become more

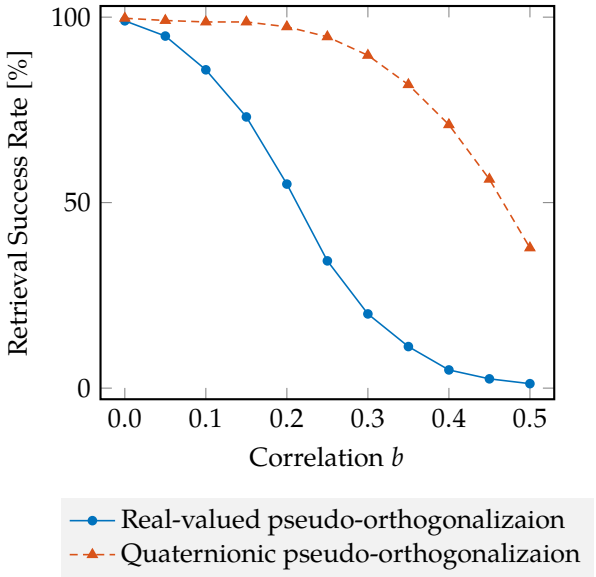


Figure 4.8: Retrieval success rates with changing the correlation in the original memory patterns. The parameters for recall process: $s = 0.9$, $\gamma = 1.002$, $\beta(0) = 1.0$.

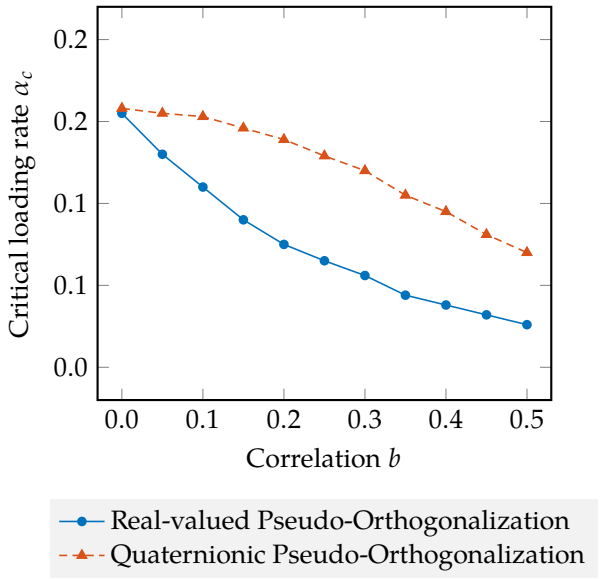


Figure 4.9: Critical loading rates with changing the correlation in the original memory patterns.

unstable with the increase of the correlation of the original memory patterns in RHNNs. In contrast, the memory patterns in QHNNs are stable when the correlation is increased. Therefore, the retrieval performance of QHNNs are maintained even if the correlation in the memory patterns is increased.

4.8 Conclusion

In this paper, we have investigated the embedding and retrieval performances for quaternionic extensions of the pseudo-orthogonalization scheme.

The numerical results show that the proposed scheme can embed the patterns better than the conventional (real-valued) scheme from the viewpoint of loading rates. This seems due to the improvement of orthogonalization achieved by the degree of freedom on the dimensions in quaternion number systems.

we have also investigated the stability and retrieval performances for the proposed scheme from the viewpoint of correlations in memory patterns. The experimental results show that the pseudo-orthogonalized patterns tend to be more unstable with the

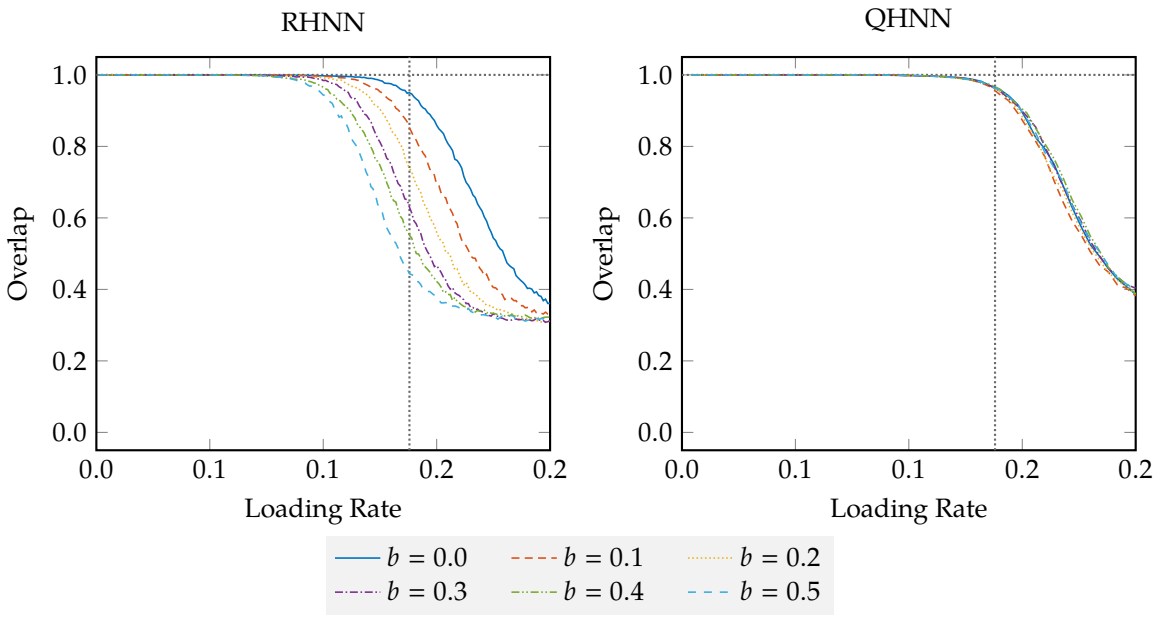


Figure 4.10: The stability of the pseudo-orthogonalized patterns in Hopfield neural networks.

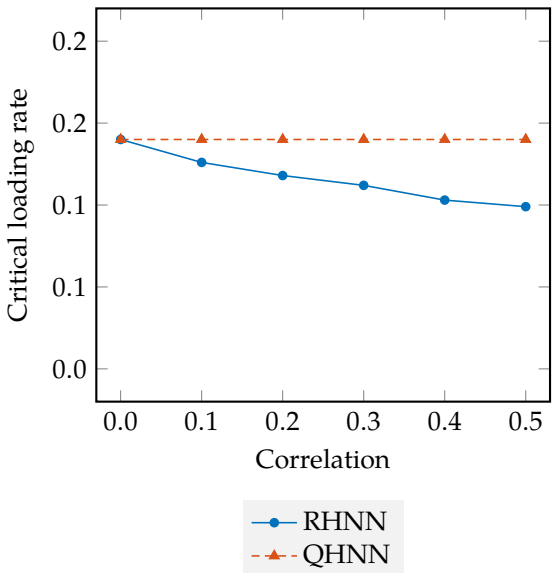


Figure 4.11: The critical loading rate against the correlation of the original memory patterns.

increase of the correlation in the original memory patterns in conventional real-valued pseudo-orthogonalization scheme. In contrast, the patterns by using the extended pseudo-orthogonalization are stable even if the correlation of memory patterns is increased. Thus, the extended scheme can stabilize highly correlated memory patterns better than conventional real-valued one. On retrieving the stored patterns from an external stimulus pattern, the performance of the extended pseudo-orthogonalization scheme is maintained compared to the real-valued scheme under the condition of high loading rate and strong correlation in the memory patterns. This is because that the memory patterns stored by the proposed scheme are stable even if the correlation in memory patterns is increased.

Parameter dependencies for the storing and retrieval performances, such as the strength of input stimuli on the retrieval stage, should be explored in detail. Also, it is important to investigate the structure on basins of attractors for quaternionic networks, as compared to the real-valued networks. These remain for our future work.

Chapter 5

Feed Forward Neural Network with Random Quaternionic Neurons

A quaternionic extension of feed forward neural network, for processing three or four dimensional signals, is proposed. This neural network is based on the three layered network with random weights, called Extreme Learning Machines (ELMs), in which iterative least-mean-square algorithms are not required for training networks. All parameters and variables in the proposed network are encoded by quaternions and operations among them follow the quaternion algebra. Neurons in the proposed network are expected to operate multi-dimensional signals as single entities, rather than real-valued neurons deal with each element of signals independently. The performances for the proposed network are evaluated through two types of experiments: classifications and reconstructions for color images in the CIFAR-10 dataset. The experimental results show that the proposed networks are superior in terms of classification accuracies for input images than the conventional (real-valued) networks with similar degrees of freedom. The detailed investigations for operations in the proposed networks are conducted.

5.1 Introduction

Processing multi-dimensional signals, such as color images, is an important problem in artificial neural networks. Artificial neural networks consist of many neurons, interconnected to each other, that accept only real-valued signals for their input, internal states, and output. Of course, these neural networks cope with high dimensional signals by configuring neurons so that each of them covers each element in these signals. But this type of configuration would be unnatural because each of elements in multi-dimensional signals is not independent to each other and these signals should be processed as a single entity. Thus, for over two decades, applications of complex values to neural networks have been extensively investigated, as summarized in the references [32, 33, 34]. Besides these studies, neural networks with dimensions more than two have also been explored: one motivation is inspired by a natural extension from real-valued neural networks to complex-valued ones. Another motivation and necessity arise from engineering applications in which multi-dimensional signals, such as three-dimensional components in color images (red, green, and blue) or body coordinates in three dimensional space (X, Y, Z), should be processed. Although neural networks for these applications can be composed by real-valued or complex-valued neurons, it would be useful to introduce a number system with high dimensions, the so-called hypercomplex number systems.

Quaternion is a four-dimensional hypercomplex number system introduced by W. R. Hamilton [10, 35]. This number system has been extensively employed in the fields of modern mathematics, physics, control of satellites, computer graphics, signal processing, and so on [36, 37, 38, 39, 40]. One of the benefits provided by quaternions is that operators in quaternions efficiently accomplish the affine transformations in three-dimensional space, especially spatial rotations, with their compact representations. Thus, it is expected that neurons with quaternionic representation and operations would be useful for processing three- and four-dimensional signals.

Feed forward neural networks, or multilayer perceptron (MLP) neural networks,

are most popular neural networks where input-output relations can be constructed by adjusting the connection weights in the network. Adjusting process is also called learning and typically incorporates least-mean-square (LMS) method. Feed forward neural network models based on quaternions have been explored in [41, 42, 9, 43, 44, 45, 46, 47]. Applications of quaternionic neural networks and quaternionic LMS have also been explored, such as control problem [42], signal classification [43], color image processing [9, 48], prediction of chaotic time series [49, 50, 44, 46], forecasting three-dimensional wind signals [44, 46]. Error back-propagation algorithms for training neural networks and LMS methods need to calculate gradients in the error surface in multi-dimensional space, spanned by connection weights or filter coefficients, in order to minimize the output error. In quaternionic domain, it is necessary to develop gradient operators with satisfying analytic (differentiable) conditions, as in the domains of real values or complex values. Despite the Cauthy-Riemann-Fueter (CRF) equation claims that only linear functions or constants are analytic in the quaternionic domain, other classes of analytic conditions and derivatives have been developed [51, 52, 45, 53].

Recently another type of feed forward neural networks with their training algorithms have been a focus of attention [54, 55, 56]. Called Extreme Learning Machine (ELM), this type of networks does not require gradient-base iterative LMS methods for their training. ELMs are typically three-layered networks, i.e., one input layer with neurons that accept external signals, one output layer with neurons in which output signals can be obtained, and one hidden layer with neurons that interconnect neurons in the input layer and ones in the output layer. Some parts of connection weights in the network are fixed randomly and remaining weights are calculated by solving a so-called least squares optimization problem. This one-shot learning scheme has advantages of fast calculation and of no gradient operators. There are several kinds of extensions investigated, such as many layered network (so-called deep learning) [57], ELM autoencoder [58, 59], regularized ELM [60], time series prediction [61], and complex-valued network [62].

Motivated by the work for complex-valued ELM [62], we present a quaternionic

extension of ELM, called Q-ELM, in this paper. Our Q-ELM is a three-layered network where all parameters, such as inputs, outputs, and connection weights, are encoded by quaternions, and operators in calculating neurons' states follow quaternion algebra. The basic structure of the proposed Q-ELM is the same as the existing quaternionic multilayer perceptron networks [42], and also has similar features in quaternionic adaptive filters and quaternionic Kalman filters [63]. Also, the proposed network resembles so-called quaternionic echo state network [64] for the point that a part of connection weights are randomly chosen and the rest of weights is determined by learning algorithms. But the proposed Q-ELM does not utilize gradients along the error surface produced by the networks, like error back-propagation algorithm. The performances of our Q-ELM are evaluated through the classification and autoencoding tasks on CIFAR-10 dataset[65]. These evaluations include the comparisons with the conventional (real-valued) ELMs and the quaternionic multilayer perceptron network with a gradient-based learning algorithm [42].

This chapter is organized as follows. In Section 5.2, we first recapitulate the conventional ELM model and its learning algorithm. Quaternionic extension of ELM, Q-ELM, is described in Section 5.3. Two types of experimental results on CIFAR-10 dataset are given in Section 5.4. In Section 5.5, we discuss the superior performances for Q-ELM with the results for several tasks. We finish with conclusions in Section 5.6.

5.2 Extreme Learning Machine

We first recapitulate a learning algorithm for single hidden layer feed forward neural networks called the Extreme learning machine (ELM) [54, 55]. ELM is a layered network with three layers, i.e., one input layer, one hidden layer, and one output layer. Each layer has neurons. The output of a neuron in the input layer connects to the neurons' inputs in the hidden layer and the output of a neuron in the hidden layer connects to the neurons' inputs in the output layer. There are no connections among neurons within each layer.

We assume the network trains a series of N samples $\{\boldsymbol{x}_i, \boldsymbol{t}_i\}, i = 1, 2, \ldots, N$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ is a d-dimensional real-valued input signal and $\boldsymbol{t}_i \in \mathbb{R}^m$ is an m-dimensional real-valued signal that is the desired output signal for the input signal. The output of ELM, denoted by $\boldsymbol{y}_i \in \mathbb{R}^m$, is given as

$$\boldsymbol{y}_i = \sum_{j=1}^{L} f(\boldsymbol{x}_i, \boldsymbol{w}_j, b_j)\boldsymbol{\beta}_j, \quad i = 1, \ldots, N, \tag{5.1}$$

where $\boldsymbol{w}_j \in \mathbb{R}^d$ is a set of connection weights between the j-th neuron in the hidden layer and each neuron in the input layer, b_j is the bias for j-th neuron in hidden layer, and $\boldsymbol{\beta}_j = \left[\beta_{j1}, \beta_{j2}, \ldots, \beta_{jm}\right]^{\mathrm{T}}$ is a set of connection weights between the j-th neuron in the hidden layer and each neuron in the output layer. The function $f(\cdot)$ denotes a nonlinear activation function for neurons in the hidden layer, such as a sigmoidal function given by

$$f(\boldsymbol{x}, \boldsymbol{w}, b) = \frac{1}{1 + \exp(-\boldsymbol{w} \cdot \boldsymbol{x} + b)}. \tag{5.2}$$

Namely, the output of ELM is a weighted sum of L non-linear mapping for the weighted input signals. A network with desired input-output relations according to training samples can be obtained by appropriately configuring connection weights.

A learning algorithm for ELM, i.e., a scheme for setting connection weights, is described. The connection weights $\boldsymbol{w}$ and the biases b in ELM are determined by uniform random values at the first stage and never changed, thus the learning is accomplished by setting the values in $\boldsymbol{\beta}$ to produce desired output signals. This can be formulated by solving the following least squares norm minimization problem:

$$\min_{\boldsymbol{B}} \ \|\boldsymbol{H}\boldsymbol{B} - \boldsymbol{T}\|^2, \tag{5.3}$$

where

$$\boldsymbol{H} = \begin{bmatrix} f(\boldsymbol{x}_1, \boldsymbol{w}_1, b_1) & \ldots & f(\boldsymbol{x}_1, \boldsymbol{w}_L, b_L) \\ \vdots & \ddots & \vdots \\ f(\boldsymbol{x}_N, \boldsymbol{w}_1, b_1) & \ldots & f(\boldsymbol{x}_N, \boldsymbol{w}_L, b_L) \end{bmatrix}_{N \times L}, \tag{5.4}$$

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{\beta}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{\beta}_L^{\mathrm{T}} \end{bmatrix}_{L\times m}, \ \boldsymbol{T} = \begin{bmatrix} \boldsymbol{t}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{t}_N^{\mathrm{T}} \end{bmatrix}_{N\times m}, \tag{5.5}$$

and $\|\cdot\|$ denotes the Frobenius norm. The problem in Eq. (5.3) is convex and thus the optimal solution $\hat{\boldsymbol{B}}$ is given by

$$\hat{\boldsymbol{B}} = \boldsymbol{H}^{\dagger}\boldsymbol{T}, \tag{5.6}$$

where $\boldsymbol{H}^{\dagger}$ indicates the Moore-Penrose pseudo inverse matrix of $\boldsymbol{H}$. There are several methods to calculate $\hat{\boldsymbol{B}}$, such as orthogonal projection method and singular value decomposition. In this paper, we employ a closed-form solution which is defined as follows:

$$\hat{\boldsymbol{B}} = \begin{cases} \boldsymbol{H}^{\mathrm{T}}\left(\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}\right)^{-1}\boldsymbol{T} & (N \le L) \\ \left(\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}\right)^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{T} & (N > L) \end{cases}. \tag{5.7}$$

When an ELM is used for classification problems, it is necessary to encode classes as desired output signals with respect to input signals. In this paper, m neurons are prepared in the output layer for a multiclass classification problem with m classes. In the case of the input signal $\boldsymbol{x}_i$ that belongs to the class $k_i \in \{1, \ldots, m\}$, the encoded desired output signal $\boldsymbol{t}_i$ is given as

$$\boldsymbol{t}_i = (t_{i1}, \ldots, t_{im})^{\mathrm{T}}, \quad t_{ij} = \begin{cases} 1 & j = k_i \\ 0 & j \neq k_i \end{cases}. \tag{5.8}$$

From the inputs and their corresponding desired output signals, the network can be trained. Also, it is necessary to determine the class from the neurons' outputs in the output layer after training. In this paper, a predicted class from an input $\boldsymbol{x}_i$ is given as

$$\text{Class for } \boldsymbol{x}_i = \underset{j=1,\ldots,m}{\arg\max}\, y_{ij}, \tag{5.9}$$

where y_{ij} denotes an output signal of the j-th neuron in the output layer.

5.3 Quaternionic Extreme Learning Machine

This section presents our proposed ELM, called Quaternionic ELM (Q-ELM). Our Q-ELM is an ELM where all parameters in the network, such as input, output, and bias of a neuron, and connection weights $\boldsymbol{w}$ and bias $\boldsymbol{\beta}$, are encoded by quaternions. The input and output signals for the training samples are also encoded by quaternions. The operations in calculating neuronal variables, such as weighted sums, follow the quaternion algebra. Each of the parameters and variables is expressed as a 4-tuple of real numbers, hence, the degree of freedom for a neuron in Q-ELM is basically quadruple as the one in real-valued ELM. The activation function for Q-ELM should be implemented for accepting quaternions. In this paper, we adopt a so-called split type activation function for Q-ELM, given as

$$g(\boldsymbol{x}, \boldsymbol{w}, b) = g^{(e)}(\boldsymbol{x}, \boldsymbol{w}, b) + g^{(i)}(\boldsymbol{x}, \boldsymbol{w}, b)\boldsymbol{i} + g^{(j)}(\boldsymbol{x}, \boldsymbol{w}, b)\boldsymbol{j} + g^{(k)}(\boldsymbol{x}, \boldsymbol{w}, b)\boldsymbol{k}. \tag{5.10}$$

This means that for each quaternionic component in the input value, real-valued function is incorporated. These real-valued functions are set to identical and defined as follows:

$$g^{(*)}(\boldsymbol{x}, \boldsymbol{w}, b) = \frac{1}{1 + \exp((-\boldsymbol{w} \cdot \boldsymbol{x} + b)^{(*)})}, \quad (*) = (e), (i), (j), (k), \tag{5.11}$$

where the function $g^{(e,i,j,k)}(\cdot)$ takes a quaternion as its input and gives resp. scalar or $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$ part for the input value as its output.

To solve the problem in Eq. (5.3), we have to obtain the Moore-Penrose pseudo inverse for a quaternion matrix $\boldsymbol{H}$. Complex adjoint matrix [66, 67] is adopted to obtain the inverse of the quaternion matrix. The quaternion matrix $\boldsymbol{Q} \in \mathbb{H}^{m \times n}$ is represented by using Cayley-Dickson notation as follows:

$$\boldsymbol{Q} = \left(\boldsymbol{Q}^{(e)} + \boldsymbol{Q}^{(i)}\boldsymbol{i}\right) + \left(\boldsymbol{Q}^{(j)} + \boldsymbol{Q}^{(k)}\boldsymbol{i}\right)\boldsymbol{j} \tag{5.12}$$

$$= \boldsymbol{A} + \boldsymbol{B}\boldsymbol{j}, \tag{5.13}$$

where $\boldsymbol{A} = \boldsymbol{Q}^{(e)} + \boldsymbol{Q}^{(i)}\boldsymbol{i}$ and $\boldsymbol{B} = \boldsymbol{Q}^{(j)} + \boldsymbol{Q}^{(k)}\boldsymbol{i}$ are complex matrices ($\boldsymbol{A}, \boldsymbol{B} \in \mathbb{C}^{m\times n}$). The complex adjoint matrix $\boldsymbol{Q}_e \in \mathbb{C}^{2m\times 2n}$ corresponding to $\boldsymbol{Q}$ is given by

$$\boldsymbol{Q}_e = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ -\boldsymbol{B}^* & \boldsymbol{A}^* \end{pmatrix}. \tag{5.14}$$

$\boldsymbol{Q}_e$ is a complex matrix and thus its pseudo inverse matrix can be computed by generally used algorithms, such as singular value decomposition. By using this technique, the Moore-Penrose pseudo inverse of a quaternion matrix can be computed via its complex adjoint matrix. Hence, the complex adjoint matrix $\hat{\boldsymbol{B}}_e \in \mathbb{C}^{2L\times 2m}$ for the weight matrix $\boldsymbol{B} \in \mathbb{H}^{L\times m}$ is defined by

$$\hat{\boldsymbol{B}}_e = \begin{cases} \boldsymbol{H}_e^* \left(\boldsymbol{H}_e \boldsymbol{H}_e^*\right)^{-1} \boldsymbol{T}_e & (N \le L) \\ \left(\boldsymbol{H}_e^* \boldsymbol{H}_e\right)^{-1} \boldsymbol{H}_e^* \boldsymbol{T}_e & (N > L) \end{cases}, \tag{5.15}$$

where $*$ denotes Hermitian transpose, and $\boldsymbol{H}_e \in \mathbb{C}^{2N\times 2L}$ and $\boldsymbol{T}_e \in \mathbb{C}^{2N\times 2m}$ are complex adjoint matrices for $\boldsymbol{H} \in \mathbb{H}^{N\times L}$ and $\boldsymbol{T} \in \mathbb{H}^{N\times m}$, respectively.

The output signals should be encoded when Q-ELMs are used for classification problems, as in the case of real-valued ELMs. The desired output signal $\boldsymbol{t}_i$ for the input signal $\boldsymbol{x}_i$ in the class m is given as

$$\boldsymbol{t}_i = (t_{i1}, \ldots, t_{im})^{\mathrm{T}}, \quad t_{ij} = \begin{cases} 1 + 0\boldsymbol{i} + 0\boldsymbol{j} + 0\boldsymbol{k} & j = k_i \\ 0 + 0\boldsymbol{i} + 0\boldsymbol{j} + 0\boldsymbol{k} & j \neq k_i \end{cases}. \tag{5.16}$$

Only scalar part of the output signal is used for representing the class. Similarly, the predicted class for a given input signal is given by the output signal of neurons as

$$\text{Class for } \boldsymbol{x}_i = \underset{j=1,\ldots,m}{\arg\max}\, {y_{ij}}^{(e)}, \tag{5.17}$$

where ${y_{ij}}^{(e)}$ denotes the scalar part of the output value for the j-th neuron in the output layer.

5.4 Experiments

In order to investigate the performance of the proposed Q-ELM, we perform two types of experiments: classification and autoencoding task on CIFAR-10 dataset [26]. We also

perform the same experiments by using the real-valued ELMs (R-ELMs) for comparisons. The CIFAR-10 dataset consists of 60,000 natural images in total: 50,000 images are used for training a classifier and the rest of 10,000 images are used for testing the performance of the classifier. Each image is composed of 32×32 pixels each of which is represented by 8 bit $\times$ 3 (RGB) channel color values. Images are categorized into 10 different classes and each of them has its own label to represent its class. In the classification task, the networks are trained so that the correct classes for input images can be produced by using 50,000 training images, and then classification for the 10,000 test images is conducted. In the autoencoding task, the networks accept all the pixel values for each image as their input and the same pixel values are to be output. The networks are trained by using training images and the performances are evaluated by the differences between the test input image and its output image from the trained networks. As described in the Sections 5.2 and 5.3, the training of the networks is conducted by setting the connection weights $\boldsymbol{w}$ and the biases b to random values and by setting the connection weights $\boldsymbol{\beta}$ by solving the least squares norm minimization. We use uniformly distributed random values in the range $[-1, 1]$ for setting the elements in $\boldsymbol{w}$ and b.

Before investigating the experimental results, we define several notations and quantities for evaluating the R-ELMs and Q-ELMs. First notation is the network configuration. The structure of the networks for both of R-ELMs and Q-ELMs is a three-layered network, thus the numbers of neurons for these layers can determine the structure. The configuration of the network is denoted by a 3-tuple d-l-m, where d, l, and m represent the numbers of neurons in the input layer, the hidden layer, and the output layer, respectively. We introduce the following two measures for the evaluations of the networks.

- **The total number of parameters (TNP):** TNP is calculated by a sum of the number of randomly fixed and trainable parameters in the network, i.e., the biases in the hidden layer, and the connection weights between the input and the hidden layer and between the hidden and the output layer. This measure reflects the degrees

of freedom in the network. For the R-ELM and Q-ELM, these can be respectively defined as follows:

$$\mathrm{TNP_R} = l(d + m + 1), \tag{5.18}$$

$$\mathrm{TNP_Q} = 4l(d + m + 1). \tag{5.19}$$

If the network configurations for R-ELM and Q-ELM are identical, TNP for Q-ELM is quadruple to that for R-ELM.

- **Root mean squared error (RMSE):** An RMSE represents the error between the output signal from the network and the desired output. The RMSE for R-ELMs is defined by

$$\mathrm{RMSE_R} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(t_i - y_i\right)^2}, \tag{5.20}$$

where t_i is the desired output value, and y_i is the output value from the network. In Q-ELMs, the network error should be calculated for each part of a quaternionic signal. The RMSE for Q-ELMs is defined by

$$\mathrm{RMSE_Q} = \sqrt{\frac{1}{4m}\sum_{i=1}^{m}\left(t_i - y_i\right)\left(t_i - y_i\right)^*}. \tag{5.21}$$

5.4.1 Classification task

We first explore the classification task by ELMs. In this task, a input for the network is a set of pixel values in the image, i.e., $32 \times 32 \times 3$ dimensional real-valued vector whose values are normalized in the range of $[0, 1]$. The output for the network is the class label corresponding to the input, represented as a 10-dimensional vector.

For R-ELMs, the number of neurons in the input layer is set to 3072, that is the same as the dimension of an input vector, and the number of neurons in the output layer is set to 10. The number of neurons in the hidden layer becomes a parameter, denoted by l_R, in this experiment. Thus, the structure of the network for R-ELMs is represented as 3072-l_R-10, and its TNP value is calculated by Eq. (5.18). In Q-ELMs one neuron

can accept the RGB color channels for a pixel, so the number of neurons for the input layer is 1024(= 32×32). A pixel value with RGB color channel, represented as 3-tuple (r, g, b), is encoded to a quaternion without a scalar part, i.e., $r\boldsymbol{i} + g\boldsymbol{j} + b\boldsymbol{k}$. The number of neurons in the output layer in Q-ELMs is 10. The structure of the network in Q-ELMs is denoted by 1024-l_{Q}-10, where l_{Q} denotes the number of neurons in the hidden layer.

Figure 5.1 shows the classification accuracies on CIFAR-10 training dataset (50,000 images) and test dataset (10,000 images) for R-ELM and Q-ELM with respect to TNP values. In this experiment, the parameter l_{R} is set to values from 50 to 1000 in steps of 50, and the parameter l_{Q} is adjusted so that calculated $\mathrm{TNP}_{\mathrm{Q}}$ is equal to or less than $\mathrm{TNP}_{\mathrm{R}}$. This is intended to comparing R-ELM and Q-ELM at the same (or similar) degrees of freedom in the networks. From this figure, the classification accuracies obtained by Q-ELMs are always higher than those by R-ELMs. Table 5.1 shows two examples of the classification accuracies for R-ELMs and Q-ELMs with their l_{R}, l_{Q}, and their TNP values. This result shows that the classification accuracy by a Q-ELM with $l_{\mathrm{Q}} = 409$ neurons is nearly equal to that by an R-ELM with $l_{\mathrm{R}} = 1000$ neurons. Hence, Q-ELMs with less hidden neurons achieve higher classification accuracy than R-ELMs achieve. This result also shows that at similar TNP ($l_{\mathrm{R}} = 1000$ and $l_{\mathrm{Q}} = 744$) the classification accuracy by Q-ELM is about 2.5 points higher than that by R-ELM, as also shown in Fig. 5.1.

5.4.2 Autoencoding task

We next investigate the performances for the autoencoding task by ELMs. In this experiment, each of color image in CIFAR-10 is used as an input and the same image is also used as the desired output. The networks are trained so that the original input should be reconstructed at the output layer. The structure of the network is symmetric for a network with three layers. Hence, the configurations of the networks for R-ELMs and Q-ELMs are set to 3072-l_{R}-3072 and 1024-l_{Q}-1024, respectively. The values for l_{R} and l_{Q} are set in the same way to the classification task, i.e., l_{R} varies from 50 to 1000 in 50 steps and l_{Q} is set so that $\mathrm{TNP}_{\mathrm{Q}}$ values become equal to or less than $\mathrm{TNP}_{\mathrm{R}}$. The

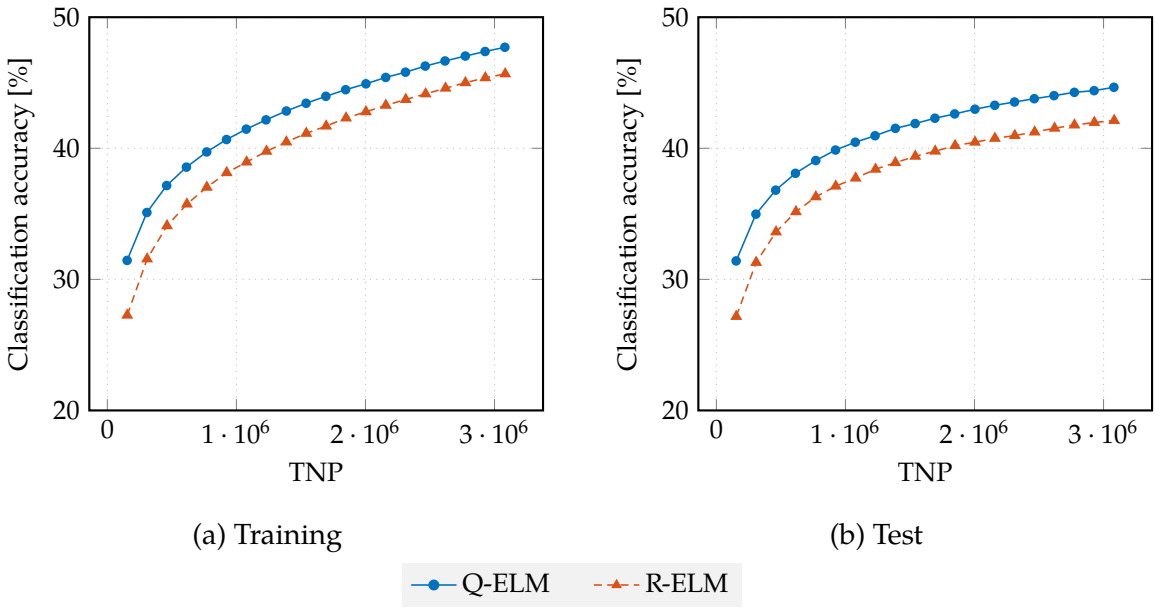


Figure 5.1: Classification accuracies for R-ELM and Q-ELM with respect to the total number of parameters (TNPs) in the networks on CIFAR10-dataset.

Table 5.1: Two examples for classification accuracy (CA) for test dataset. The network configurations for R-ELMs and Q-ELMs are 3072-l_{R}-10 and 1024-l_{Q}-10, respectively. CAs are averaged values in 100 trials with their standard deviations.

l_{R}	$\mathrm{TNP}_{\mathrm{R}}$	CA(R-ELM) [%]
550	1,695,650	39.78 ± 0.41
1000	3,083,000	42.12 ± 0.35

l_{Q}	$\mathrm{TNP}_{\mathrm{Q}}$	CA(Q-ELM)[%]
409	1,693,260	42.29 ± 0.29
744	3,080,160	44.65 ± 0.29

performance is evaluated by RMSEs calculated from the input (desired) image and the image from the network.

Figure 5.2 shows averaged RMSEs for R-ELMs and Q-ELMs with respect to TNP values. The RMSEs obtained by Q-ELMs are always smaller than those by R-ELMs, showing that Q-ELMs reconstruct more accurate images than R-ELMs. Two examples for l_{R} and l_{Q} values, TNPs, and RMSEs are shown in Table. 5.2. In this table, the averaged RMSE by Q-ELMs with $l_{\mathrm{Q}} = 449$ neurons is almost equal to that by R-ELMs with $l_{\mathrm{R}} = 1000$ neurons. This is a similar tendency to the classification task, though the tasks imposed on the ELMs are different. These results suggest that quaternionic extension of ELMs leads to compact and efficient ELM for processing multi-dimensional signals.

We also investigate the training time comparison for proposed Q-ELM and Quaternionic Multilayer Perceptron (QMLP) [50] whose training algorithm is based on gradient descent algorithm. Table 5.3 shows the training time of Q-ELM and QMLP on the autoencoding task. Mini-batch stochastic gradient descent (SGD) is employed for the training of QMLP. The sample size of mini-batch is set to 5 and the learning rate is fixed to 0.001. The RMSE and computational time of QMLP are obtained after 50,000 iterations for weight updates. We find that Q-ELM achieves better performance in RMSE than QMLP with less computational time though a learning in Q-ELM is much faster (about 12 times) than one in QMLP.

We further consider the time complexities for Q-ELM and QMLP. These orders are calculated as $O(l^2N + l^3 + dlN)$ for Q-ELM and $O(dln)$ for QMLP, where N is the number of training patterns and n is the number of training iterations for QMLP. By using the parameters in the autoencoding task, we obtain a ratio for these time complexities similar to the one for training times in Table 5.3.

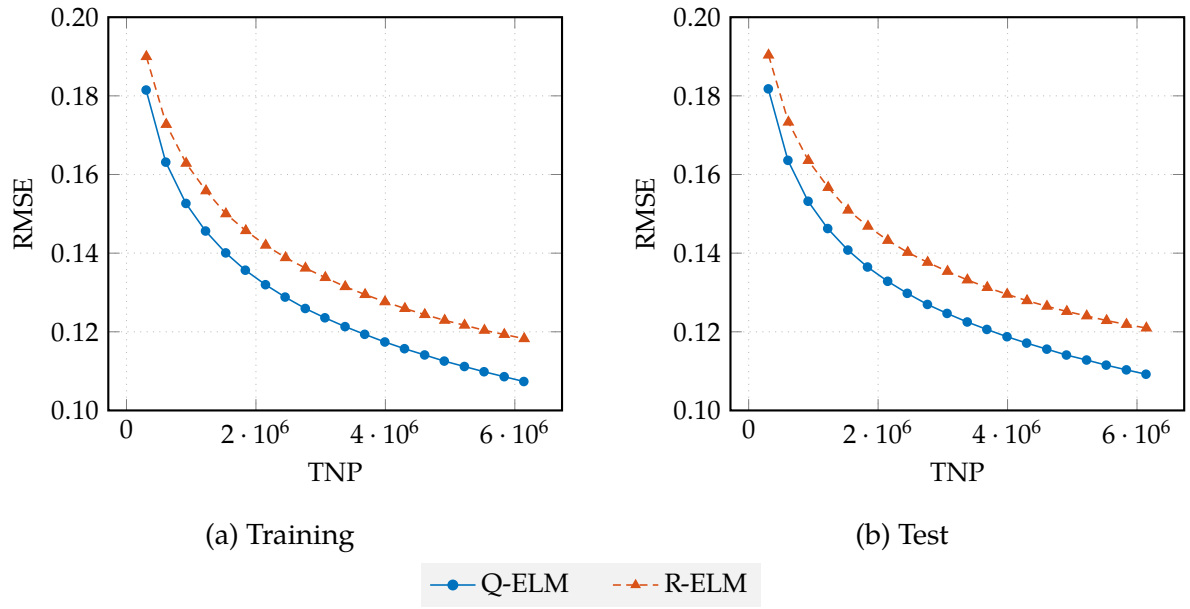


Figure 5.2: RMSEs for R-ELM and Q-ELM with respect to the total number of parameters (TNPs). Each RMSE is averaged in 100 trials.

Table 5.2: Two examples for autoencoding performances by R-ELM and Q-ELM. The network configurations for R-ELMs and Q-ELMs are 3072-l_{R}-3072 and 1024-l_{Q}-1024, respectively. Each RMSE is averaged from 100 trials for training and evaluation.

l_{R}	$\mathrm{TNP}_{\mathrm{R}}$	RMSE(R-ELM)
600	3,687,000	0.131± 0.0133
1000	6,145,000	0.121± 0.0123

l_{Q}	$\mathrm{TNP}_{\mathrm{Q}}$	RMSE(Q-ELM)
449	3,680,004	0.121± 0.0122
749	6,138,804	0.109± 0.0110

Table 5.3: A comparison for Q-ELM and QMLP [50] on the autoencoding task. All experiments are carried out on a PC with Intel Core i7-6700K 4.00 GHz CPU and 64 GB RAM.

Model	l_{Q}	RMSE	Training time [s]
Q-ELM	749	0.109	1,056
QMLP	749	0.115	12,510

5.5 Discussion

In Section 5.4, we have shown that Q-ELMs achieve better classification accuracies and more accurate reconstruction than R-ELMs on the tasks using CIFAR-10 dataset. In this section, we explore the reasons why Q-ELMs have superior performances through distributions of neuronal outputs, simple transformation tasks, and detailed analysis on images produced by ELMs.

First, we focus on the computational abilities of a quaternionic neuron with respect to the one of real-valued neurons. We evaluate them by obtaining output distributions for single quaternionic and real-valued neurons. A quaternionic neuron with its input weight is prepared, and the outputs of this neuron are obtained by a fixed quaternionic input $x = (0, 1, 1, 1)^{\mathrm{T}}$, fixed bias $(0, 0, 0, 0)^{\mathrm{T}}$, and various weight values. A weight value $w = (w_0, w_1, w_2, w_3) \in \mathbb{R}^4$ is set to a random quaternionic value with each of quaternionic component (w_0, w_1, w_2, w_3) being in the range $[-1, 1]$. The output of a neuron is calculated by $y = g(x, w, 0)$. 10^9 weight values are used for obtaining the neurons' outputs. The same experiment is conducted on real-valued neurons. For aligning the dimension of a quaternionic neuron, four input and output neurons with 16 connection weights are prepared. An input vector for these neurons is set to $(0, 1, 1, 1)^{\mathrm{T}}$, the biases of neurons are set to 0, and each of weight values is set to a random value in the range $[-1, 1]$. There are also 10^9 sets of weight values for processing in the real-valued neurons.

Figure 5.3 shows distributions of neuron's output, where all combinations of two components among the four components of quaternionic output, i.e., $y^{(e)}$, $y^{(i)}$, $y^{(j)}$, $y^{(k)}$, are used. Also, the distributions in Fig. 5.4 show all the combinations of four real-valued neurons' outputs. Comparing the distributions from quaternionic and real-valued neurons' outputs, we find that the output values from the real-valued neuron have some contraction to the center of the distribution. This means that the real-valued neuron tends to output values around zero. This tendency is slightly suppressed for the outputs from the quaternionic neuron. The numbers of effective parameters for the connection weights are 12 for a real-valued neuron and 3 for a quaternionic neuron (this is due to a zero value in the neurons' input). Hence, these distributions suggest that quaternionic neuron can produce a variety of output values with less number of connection weights, and we see that this variety can be produced by the quaternionic operators conducted in quaternionic neurons and that this can be one reason for a superior performance achieved by the proposed Q-ELM.

Next, we conduct an experiment on affine transformations in three-dimensional space by Q-ELMs and R-ELMs in order to clarify the operations in the Q-ELMs and R-ELMs. In this experiment, the relationship between two coordinates in three-dimensional space is trained by ELMs. Hence, the networks used in the experiment have a symmetrical structure, i.e., 1-l_{Q}-1, 3-l_{R}-3, where the input and output neurons process the coordinate of a point in three-dimensional space. This experiment consists of three types of affine transformation tasks, described as follows:

- **Translation:**
  The input plane should be shifted by -0.3 along the z axis.
- **Scaling:**
  The input plane should be enlarged twice at each side.
- **Rotation:**
  The input plane should be rotated in 45 degrees around the y axis.

Figure 5.5 shows the input and desired output data points of the training dataset for

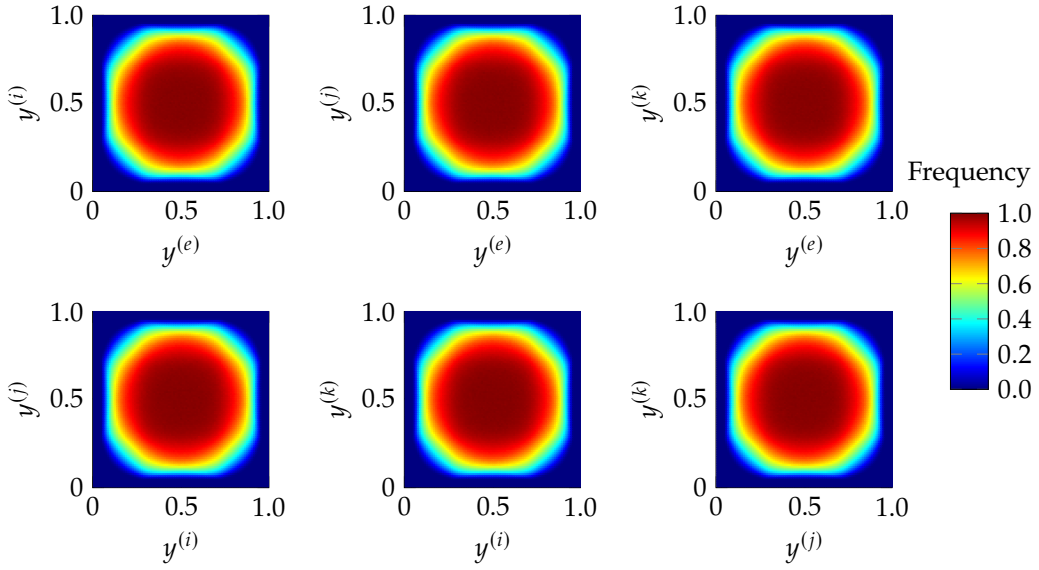


Figure 5.3: 2D histograms for the distributions of quaternionic neuron output.

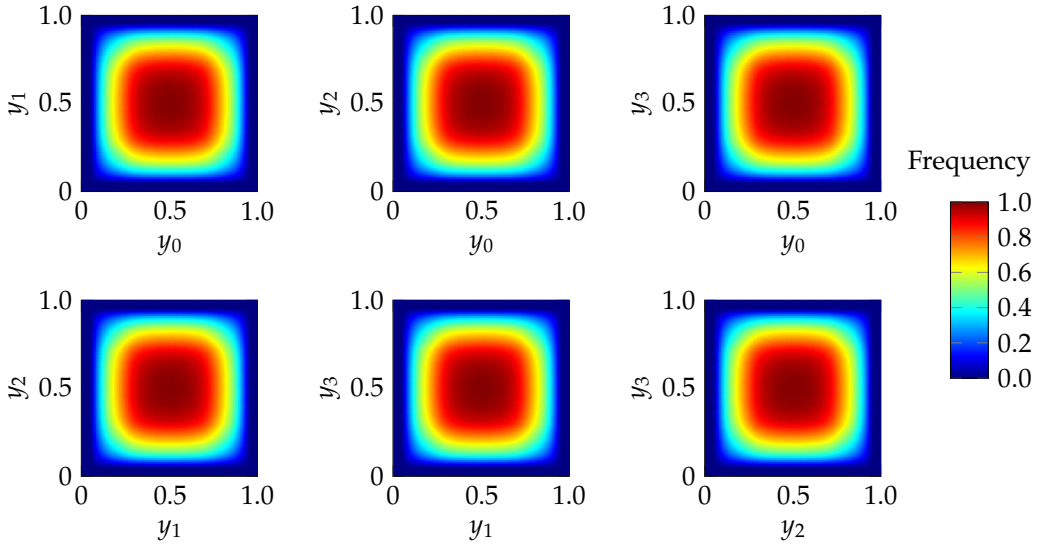


Figure 5.4: 2D histograms for the distributions of real-valued neurons' outputs.

these tasks, where the inputs are indicated as blue points and the desired outputs are indicated as red points. Both input dataset and desired output dataset consist of 9 points that constitute planes in three-dimensional space. For evaluating the generalization performances, test input points shown in Fig. 5.6 are input to the networks after the training process.

Figures 5.7 and 5.8 show examples of the scatter plot for test inputs and outputs data from networks in three-dimensional space, for Q-ELM and R-ELM, respectively. In these examples, a Q-ELM with 1-8-1 configuration ($\mathrm{TNP_Q} = 96$) and an R-ELM with 3-15-3 configuration ($\mathrm{TNP_R} = 105$) are used. We see from these figures that the Q-ELM can transform the test inputs into the outputs with maintaining their three-dimensional structure, whereas the R-ELM fails. The transformations performed by this R-ELM seem to be projections to the desired output points in training processes rather than affine transformations from input and desired output points. We perform the same experiment with various l_{Q} and l_{R} values. Figure 5.9 shows RMSEs with respect to TNP values for Q-ELMs and R-ELMs, for the experiments of three types of affine transformations. The RMSEs calculated by training data are shown in Figs. 5.9(a), 5.9(c), and 5.9(e), and those by test data are shown in Figs. 5.9(b), 5.9(d), and 5.9(f). From these figures, it is shown that both types of networks can grasp the input-output relationships in the training dataset but R-ELMs cannot get operations even if they have enough degrees of freedom in them. Q-ELMs have quaternionic representations and also operations in then, thus they can deal with three- or four-dimensional signals as single entities and these signals can be operated in affine transformation manners. This can be a key to achieve better performances by Q-ELMs than those by R-ELMs with similar degrees of freedom in which each element in multi-dimensional data has to be processed individually in the real-valued neurons.

Finally, we examine the detailed output images in the autoencoding task in Section 5.4.2, from the view point of chromatic information for the reconstructed images by ELMs. Figure 5.10 shows three examples of reconstructed images with their chromatic plots. The chromatic plot in this figure is a scatter plot for pixel values for all

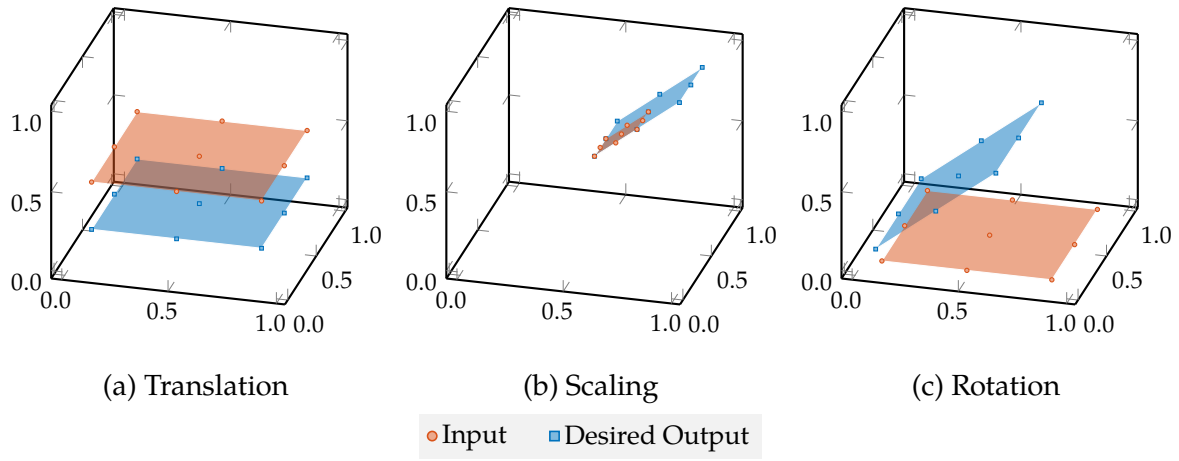


Figure 5.5: Training data points for affine transformation tasks. Both input and desired output consist of 9 points. Each of them constitutes a plane in three-dimensional space.

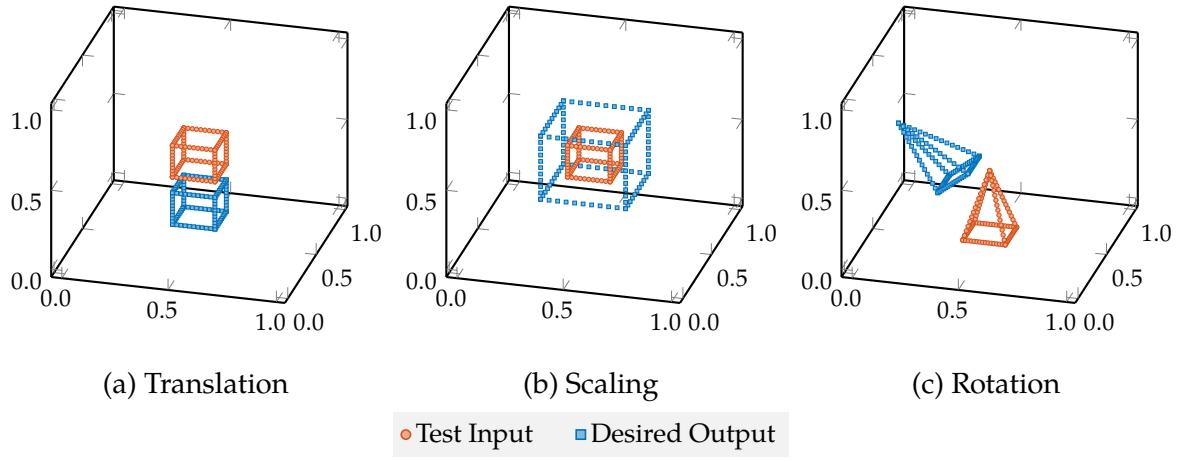
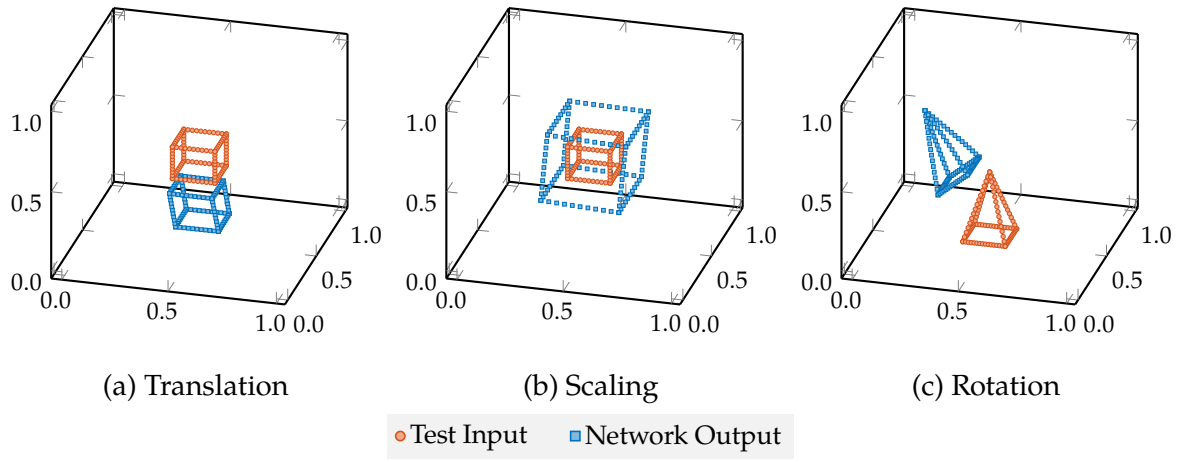
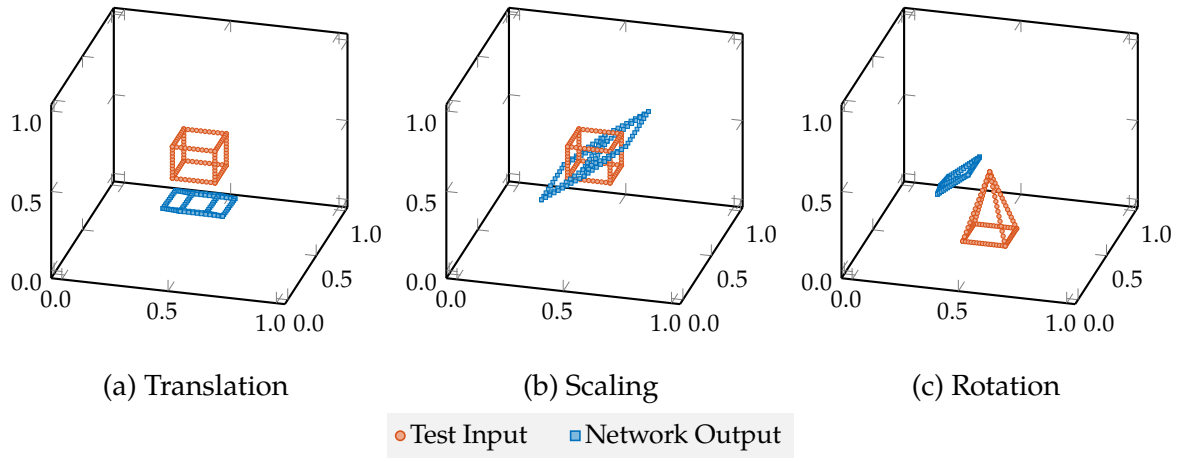


Figure 5.6: Test input points and desired output data points.



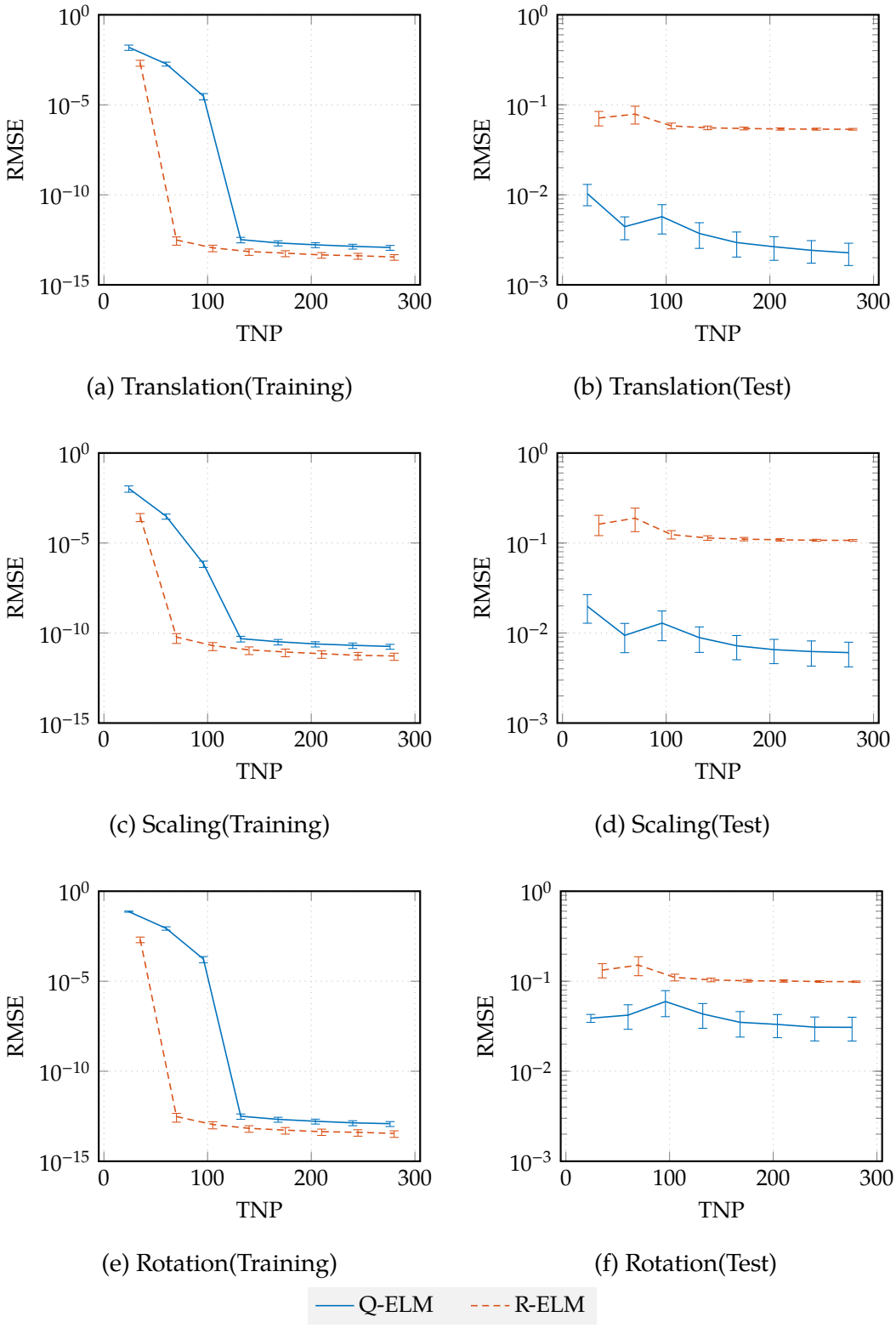
(a) Translation (b) Scaling (c) Rotation

Figure 5.7: Test input points and output data points obtained by Q-ELM with 1-8-1 configuration.



(a) Translation (b) Scaling (c) Rotation

Figure 5.8: Test input points and output data points obtained by R-ELM with 3-15-3 configuration.



(a) Translation(Training)

(b) Translation(Test)

(c) Scaling(Training)

(d) Scaling(Test)

(e) Rotation(Training)

(f) Rotation(Test)

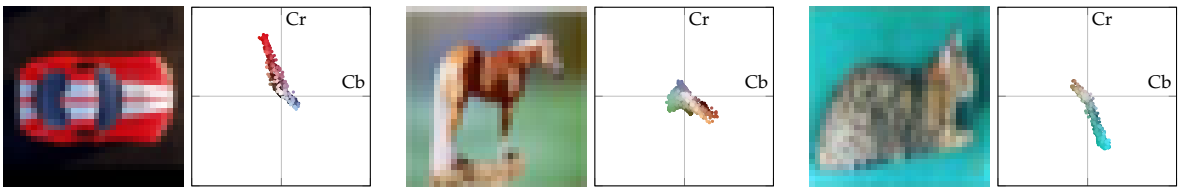
Figure 5.9: RMSE calculated from training and test dataset with respect to the total number of parameters (TNPs) in affine transformation task.

pixels in an image, where pixel values are represented in YCbCr color space and the values of Cb and Cr for pixels are used in the plot. Thus, this plot indicates a color distribution in an image regardless of intensities for pixels. The reconstructed images are obtained from a Q-ELM with $l_{\mathrm{Q}} = 749$ and an R-ELM with $l_{\mathrm{R}} = 1000$. From the images in Fig. 5.10, we see that the distributions by Q-ELMs can cover those of the original (input) images, while the distributions by R-ELMs do not fully cover them. This suggest that R-ELMs cannot reconstruct a part of color space by their projection-based operations. On the other hand, Q-ELMs can produce a wide variety of colors by their affine transformation-based operations with pixel colors being processed as single entities. The operations of signals in Q-ELMs and R-ELMs will make differences for their performances for the image reconstructing task.

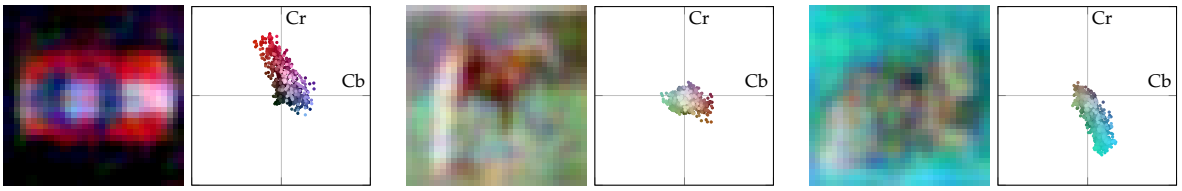
5.6 Conclusion

In this chapter, we proposed a quaternionic extension of Extreme Learning Machine, called Q-ELM, and investigated its performances as compared with conventional (real-valued) ELMs. A Q-ELM is a layered neural network with three layers of neurons, i.e., the input layer, hidden layer, and output layer. All parameters and variables in a Q-ELM network are encoded by quaternions and operations among them follow the quaternion algebra. The connection weights between neurons in the input and hidden layers are randomly fixed, and training the network is formulated as a least squares norm minimization problem. Thus, this type of networks has advantages to the neural networks with least-mean-square method in the points of less computation cost and no gradient operators.

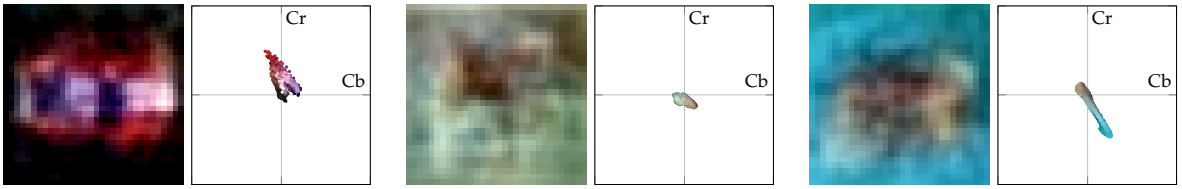
For evaluating the performances of Q-ELMs, we performed two types of experiments by using CIFAR-10 image dataset: the classification of images and autoencoding tasks. It is shown from the experimental results that Q-ELMs have better classification and reconstruction abilities than real-valued ELMs. We also investigated detailed operations in Q-ELMs through conducting simple transformation tasks and analysis of re-



(a) Desired output (test input)



(b) Output from Q-ELM



(c) Output from R-ELM

Figure 5.10: Three examples of output images with their chromatic distributions in autoencoding task. A chromatic distribution for an image is shown by a scatter plot in which each point for a pixel is placed at its Cb and Cr components as its coordinate. For each example, left-hand side shows an original image or a reconstructed image from the network and right-hand side shows its chromatic distribution. The configurations of networks are 1024-749-1024 for Q-ELM and 3072-1000-3072 for R-ELM.

constructed images. These investigations show that Q-ELMs process three-dimensional signals as single entities with affine transformations. On the other hand, real-valued ELMs with many degrees of freedom operate each of elements in multi-dimensional signals by each of neurons independently. Hence, for processing multi-dimensional signals such as color images or body coordinates, Q-ELMs can serve superior abilities.

There are a lot of ELMs available with extensions of many-layer network for deep learning [57], convolutional networks [68], and so on. For achieving better performances for natural image processing, incorporating these extensions are inevitable challenges for us. Low computational cost will be useful for practical applications with embedding processors, such as autonomous vehicles and robots. Image processing and recognitions for these applications will be for our future work.

More analysis on the reason why superior performances can be achieved by Q-ELM should be performed from both of theoretical and experimental viewpoints. Incorporating the theoretical investigations will be useful; for an example, computational power for single quaternionic neuron, where we have conducted in section 5, have been theoretically explored in [69].

Echo State Networks (ESNs) are recurrent neural networks recently proposed [70, 71], which have similar architecture to ELMs. These networks have applications of chaotic time-series predictions [72] due to their powerful prediction ability for complex time-series signals. Both of ELMs and ESNs have a set of weights, and a partial set of weights is randomly fixed and remaining weights are determined by learning algorithms, typically by solving a least squares optimization problem. This type of networks also does not require gradient-based algorithm as in the cases of ELMs. Extensions for ESNs have been widely investigated and its quaternionic extension is available with its practical applications [64]. There have been several researches concerning the comparisons between real-valued ESN and ELM [61], so the performance comparisons between the quaternionic ESN and the proposed Q-ELM under time-series prediction problems would be interesting challenges.

Chapter 6

Conclusions

The importance of neural network research has been increasing in recent years with the revitalization of the practical application of machine learning as typified by deep learning. In this study, we discuss quaternion extensions of neural computing in such research trend.

A neural network is one of computing methods that imitates the information processing in the brain. In recent years, a method using multiple values such as complex numbers and quaternions is attracting attention for information representation of neurons constituting a network. Compared with conventional real-valued information representation, it has been shown that these hypercomplex extended models have high generalization performance in engineering problems dealing with spatial coordinates. In particular, there is little knowledge about the quaternion models. Therefore, in this study, we proposed and evaluated the several quaternionic neural network models in order to examine the characteristics and efficacy by extending the neural networks by using quaternions.

Regarding mutual connected neural networks, we mainly examined the recall performance for associative memories by using quaternions in polar form. We also clarified the main factor of the performance degradation of quaternionic Hopfield associative memory (QHAM) is rotation invariance of the memory patterns throgh several experiments. In addition, we proposed another types of quaternionic associative memory

models for improving the performance of the QHAM. One is quaternionic bipartite auto-associative memory (QBAAM), and the other is quaternionic Hopfield associative memory with dual connections (QHAMDC). QBAAM and QHAMDC can reduce the spurious patterns caused by the rotation invariance by combining real-valued neurons and quaternionic neurons or utilizing non-commutative property of quaternions. By using these models, memory patterns such as natural images which have high bit depth signals can be embedded to and retrieved from the network.

As for the multilayered neural networks, we propose a quaternionic extension of extreme learning machines (ELMs). Classification and autoencoding tasks by using general object recognition benchmarks show that quaternionic ELMs have higher generalization performance than conventional real-valued ELMs. It is highly important that the learning performance of quaternionic ELMs for affine transformations in three dimensional space is superior to that of real-valued ELMs.

The above results fully demonstrate the usefulness of introducing information processing method based on quaternions to neurocomputing, and it can be expected to acceralate the development of computational intelligence by expanding the knowledge obtained in this study into deep learning.

Bibliography

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016. http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html

[2] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," *IDC iView: IDC Analyze the future*, vol. 2007, no. 2012, pp. 1–16, 2012.

[3] Top500.org. https://www.top500.org

[4] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[5] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[6] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[7] T. Nitta, "Learning transformations with complex-valued neurocomputing," *Int. J. Organ. Collect. Intell.*, vol. 3, no. 2, pp. 81–116, Apr. 2012.

[8] H. Kusamichi, T. Isokawa, N. Matsui, Y. Ogawa, and K. Maeda, "A new scheme for color night vision by quaternion neural network," in *Proceedings of the 2nd International Conference on Autonomous Robots and Agents (ICARA2004)*, 2004, pp. 101–106.

[9] N. Matsui, T. Isokawa, H. Kusamichi, F. Peper, and H. Nishimura, "Quaternion neural network with geometrical operators," *Journal of Intelligent and Fuzzy Systems*, vol. 15, no. 3–4, pp. 149–164, 2004.

[10] W. R. Hamilton, *Lectures on Quaternions*. Dublin: Hodges and Smith, 1853.

[11] T. A. Ell and S. J. Sangwine, "Quaternion involutions and anti-involutions," *Computers & Mathematics with Applications*, vol. 53, no. 1, pp. 137–143, 2007.

[12] T. Bülow, "Hypercomplex spectral signal representations for the processing and analysis of images," Ph.D. dissertation, Christian-Albrechts-Universität zu Kiel, 1999.

[13] T. Bülow and S. G., "Hypercomplex signals—a novel extension of the analytic signal to the multidimensional case," *IEEE Transactions on Signal Processing*, vol. 49, no. 11, pp. 2844–2852, 2001.

[14] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-sate neurons," *Proceedings of the National Academy of Sciences USA*, vol. 81, pp. 3088–3092, 1984.

[15] N. N. Aizenberg, Y. L. Ivaskiv, and D. A. Pospelov, "About one generalization of the threshold function," *Doklady Akademii Nauk SSSR (The Reports of the Academy of Sciences of the USSR)*, vol. 196, no. 6, pp. 1287–1290, 1971.

[16] A. J. Noest, "Discrete-state phasor neural networks," *Physical Review A*, vol. 38, no. 4, pp. 2196–2199, 1988.

[17] N. N. Aizenberg and I. N. Aizenberg, "Cnn based on multi-valued neuron as a model of associative memory for gray-scale images," in *Proceedings of the 2nd IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA-92)*, 1992, pp. 36–41.

[18] S. Jankowski, A. Lozowski, and J. M. Zurada, "Complex-valued multistate neural associative memory," *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1491–1496, 1996.

[19] H. Aoki, M. R. Azimi-Sadjadi, and Y. Kosugi, "Image association using a complex-valued associative memory model," *IEICE Transactions on Fundamental of Electronics, Communications and Computer Sciences*, vol. E83-A, pp. 1824–1832, 2000.

[20] M. K. Müezzinoğlu, C. Güzeliş, and J. M. Zurada, "A new design method for the complex-valued multistate Hopfield associative memory," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, pp. 891–899, 2003.

[21] D.-L. Lee, "Improvements of complex-valued hopfield associative memory by using generalized projection rules," *IEEE Transaction on Neural Networks*, vol. 17, no. 5, pp. 1341–1347, 2006.

[22] Y. Suzuki and M. Kobayashi, "Complex-valued bipartite auto-associative memory," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 97, no. 8, pp. 1680–1687, 2014.

[23] T. Isokawa, H. Nishimura, A. Saitoh, N. Kamiura, and N. Matsui, "On the scheme of quaternionic multistate Hopfield neural network," in *Proceedings of Joint 4th International Conference on Soft Computing and Intelligent Systems and 9th International Symposium on advanced Intelligent Systems (SCIS & ISIS 2008)*, 2008, pp. 809–813.

[24] L. Personnaz, I. Guyon, and G. Dreyfus, "Collective computational properties of neural networks: New learning mechanisms," *Phys. Rev. A*, vol. 34, pp. 4217–4228, 1986.

[25] Y. Yano and Y. Osana, "Chaotic complex-valued bidirectional associative memory," in *Proceedings of IEEE and INNS International Joint Conference on Neural Networks*, 2009, pp. 3444–3449.

[26] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[27] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79, pp. 2554–2558, 1982.

[28] S. Diederich and M. Opper, "Learning of correlated patterns in spin-glass networks by local learning rules," *Phys. Rev. Lett.*, vol. 58, pp. 949–952, 1987.

[29] M. Oku, T. Makino, and K. Aihara, "Pseudo-orthogonalization of memory patterns for associative memory," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24(11), pp. 1877 – 1887, 2013.

[30] T. Isokawa, H. Nishimura, N.Kamiura, and N.Matsui, "Associative memory in quaternionic Hopfield neural network," *International Journal of Neural Systems*, vol. 18, no. 2, pp. 135–145, 2008.

[31] T. Isokawa, N. Matsui, and H. Nishimura, "Quaternionic neural networks for associative memories," in *Complex-Valued Neural Networks: Advances and Applications*, A. Hirose, Ed. Wiley-IEEE Press, 2013, ch. 5, pp. 103–132.

[32] A. Hirose, Ed., *Complex-Valued Neural Networks: Theories and Application*, ser. Innovative Intelligence. Singapore: World Scientific Publishing, 2003, vol. 5.

[33] A. Hirose, *Complex-Valued Neural Networks*, ser. Studies in Computational Intelligence. Berlin, Heidelberg: Springer-Verlag, 2006, vol. 32.

[34] T. Nitta, Ed., *Complex-Valued Neural Networks: Utilizing High-Dimensional Parameters*. Hershey, New York: Information Science Reference, 2009.

[35] T. L. Hankins, *Sir William Rowan Hamilton*. Baltimore and London: Johns Hopkins University Press, 1980.

[36] S. G. Hoggar, *Mathematics for Computer Graphics*. Cambridge: Cambridge University Press, 1992.

[37] J. B. Kuipers, *Quaternions and Rotation Sequences: A Primer With Applications to Orbits, Aerospace and Virtual Reality*. Princeton: Princeton Univ Press, 1998.

[38] R. Mukundan, "Quaternions: From classical mechanics to computer graphics, and beyond," in *Proceedings of the 7th Asian Technology Conference in Mathematics*, 2002, pp. 97–105.

[39] E. M. S. Hitzer, "Quaternion fourier transform on quaternion fields and generalizations," *Advances in Applied Clifford Algebras*, vol. 17, no. 3, pp. 497–517, 2007.

[40] T. A. Ell and S. J. Sangwine, "Hypercomplex fourier transforms of color images," *Image Processing, IEEE Transactions on*, vol. 16, no. 1, pp. 22–35, 2007.

[41] T. Nitta, "An extension of the back-propagation algorithm to quaternions," in *Proceedings of International Conference on Neural Information Processing (ICONIP'96)*, vol. 1, 1996, pp. 247–250.

[42] P. Arena, L. Fortuna, M. Muscato, and M. G. Xibilia, "Multilayer perceptrons to approximate quaternion valued functions," *Neural Networks*, vol. 10, no. 2, pp. 335–342, 1997.

[43] S. Buchholz and N. Le Bihan, "Polarized signal classification by complex and quaternionic multi-layer perceptrons," *International Journal of Neural Systems*, vol. 18, no. 02, pp. 75–85, 2008.

[44] B. C. Ujang, C. C. Took, and D. P. Mandic, "Split quaternion nonlinear adaptive filtering," *Neural Networks*, vol. 23, no. 3, pp. 426–434, 2010.

[45] D. P. Mandic, C. Jahanchahi, and C. C. Took, "A quaternion gradient operator and its applications," *IEEE Signal Processing Letters*, vol. 18, no. 1, pp. 47–50, Jan 2011.

[46] B. C. Ujang, C. C. Took, and D. P. Mandic, "Quaternion-valued nonlinear adaptive filtering," *IEEE Transactions on Neural Networks*, vol. 22, no. 8, pp. 1193–1206, Aug 2011.

[47] T. Isokawa, H. Nishimura, and N. Matsui, "Quaternionic multilayer perceptron with local analyticity," *Information*, vol. 3, no. 4, p. 756, 2012.

[48] H. Kusamichi, T. Isokawa, N. Matsui, Y. Ogawa, and K. Maeda, "A new scheme for color night vision by quaternion neural network," in *In Proceedings of the 2nd International Conference on Autonomous Robots and Agents (ICARA2004)*, 2004, pp. 101–106.

[49] P. Arena, S. Baglio, L. Fortuna, and M. G. Xibilia, "Chaotic time series prediction via quaternionic multilayer perceptrons," in *Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on*, vol. 2, 1995, pp. 1790–1794.

[50] P. Arena, R. Caponetto, L. Fortuna, G. Muscato, and M. G. Xibilia, "Quaternionic multilayer perceptrons for chaotic time series prediction," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 79, no. 10, pp. 1682–1688, 1996.

[51] S. D. Leo and P. P. Rotelli, "Quaternonic analyticity," *Applied Mathematics Letters*, vol. 16, no. 7, pp. 1077–1081, 2003.

[52] C. Schwartz, "Calculus with a quaternionic variable," *Journal of Mathematical Physics*, vol. 50, no. 1, pp. 1–11, 2009.

[53] D. Xu and D. P. Mandic, "The theory of quaternion matrix derivatives," *IEEE Transactions on Signal Processing*, vol. 63, no. 6, pp. 1543–1556, March 2015.

[54] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 2, 2004, pp. 985–990.

[55] ——, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.

[56] G. Huang, G.-B. Huang, S. Song, and K. You, "Trends in extreme learning machines: A review," *Neural Networks*, vol. 61, pp. 32 – 48, 2015.

[57] E. Cambria, G.-B. Huang, L. L. C. Kasun, H. Zhou, C. M. Vong, J. Lin, J. Yin, Z. Cai, Q. Liu, K. Li, V. C. M. Leung, L. Feng, Y.-S. Ong, M.-H. Lim, A. Akusok, A. Lendasse, F. Corona, R. Nian, Y. Miche, P. Gastaldo, R. Zunino, S. Decherchi, X. Yang, K. Mao, B.-S. Oh, J. Jeon, K.-A. Toh, A. B. J. Teoh, J. Kim, H. Yu, Y. Chen, and J. Liu, "Extreme learning machines [trends & controversies]," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 30–59, 2013.

[58] L. L. C. Kasun, H. Zhou, G.-B. Huang, and C. M. Vong, "Representational learning with extreme learning machine for big data," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 31–34, December 2013.

[59] J. Tang, C. Deng, and G. B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 4, pp. 809–821, April 2016.

[60] G. B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513–529, April 2012.

[61] J. Butcher, D. Verstraeten, B. Schrauwen, C. Day, and P. Haycock, "Reservoir computing and extreme learning machines for non-linear time-series data analysis," *Neural networks*, vol. 38, pp. 76–89, 2013.

[62] M.-B. Li, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "Fully complex extreme learning machine," *Neurocomputing*, vol. 68, pp. 306–314, 2005.

[63] D. H. Dini, C. Jahanchahi, and D. P. Mandic, "Kalman filtering for widely linear complex and quaternion valued bearings only tracking," *IET Signal Processing*, vol. 6, no. 5, pp. 435–445, July 2012.

[64] Y. Xia, C. Jahanchahi, and D. P. Mandic, "Quaternion-valued echo state networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 4, pp. 663–673, April 2015.

[65] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[66] J. L. Brenner, "Matrices of quaternions," *Pacific J. Math*, vol. 1, pp. 329–335, 1951.

[67] F. Zhang, "Quaternions and matrices of quaternions," *Linear Algebra and its Applications*, vol. 251, pp. 21–57, 1997.

[68] G. B. Huang, Z. Bai, L. L. C. Kasun, and C. M. Vong, "Local receptive fields based extreme learning machine," *IEEE Computational Intelligence Magazine*, vol. 10, no. 2, pp. 18–29, May 2015.

[69] T. Nitta, "A solution to the 4-bit parity problem with a single quaternary neuron," *Neural Information Processing - Letters and Reviews*, vol. 5, no. 2, pp. 33–39, 2004.

[70] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, p. 34, 2001.

[71] ——, "Adaptive nonlinear system identification with echo state networks," in *Advances in neural information processing systems*, 2002, pp. 593–600.

[72] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *science*, vol. 304, no. 5667, pp. 78–80, 2004.

Author's Papers Concerning this Dissertation

Chapter 3

[1] T. Minemoto, T. Isokawa, H. Nishimura, and N.Matsui, "Extended projection rule for quaternionic multistate Hopfield neural network," in *Proceedings of 20th International Symposium on Artificial Life and Robotics 2015 (AROB2015, Japan)*, 2015, pp. 418–423.

[2] T. Minemoto, T. Isokawa, H. Nishimura, and N. Matsui, "Quaternionic multistate Hopfield neural network with extended projection rule," *Artificial Life and Robotics*, vol. 21, no. 1, pp. 106–111, 2016.

[3] T. Minemoto, T. Isokawa, N. Matsui, M. Kobayashi, and H. Nishimura, "On the performance of quaternionic bidirectional auto-associative memory," in *Proceedings of International Joint Conference on Neural Networks (IJCNN2015, Ireland)*, 2015, pp. 2910–2915.

[4] T. Minemoto, T. Isokawa, M. Kobayashi, H. Nishimura, and N. Matsui, "Pattern retrieval by quaternionic associative memory with dual connections," in *Proceedings of 23st International Conference on Neural Information Processing (ICONIP2016, Japan), LNCS9949*, 2016, pp. 317–325.

[5] T. Minemoto, T. Isokawa, N. Matsui, M. Kobayashi, and H. Nishimura, "Retrieval performance of hopfield associative memory with complex-vaued and real-valued neurons," in *Proceedings of International Joint Conference on Neural Networks (IJCNN2016, Canada)*, 2016, pp. 4133–4138.

Chapter 4

[1] T. Minemoto, T. Isokawa, H. Nishimura, and N. Matsui, "Utilizing high-dimensional neural networks for pseudo-orthogonalization of memory patterns," in *Proceedings of 21st International Conference on Neural Information Processing (ICONIP2014, Malaysia), LNCS8834*, 2014, pp. 527–534.

[2] ——, "Pseudo-orthogonalization of memory patterns for complex-valued and quaternionic associative memories," *Journal of Artificial Intelligence and Soft Computing Research*, 2016 (Accepted).

Chapter 5

[1] T. Minemoto, T. Isokawa, H. Nishimura, and N. Matsui, "Feed forward neural network with random quaternionic neurons," *Signal Processing*, 2016 (In Press).

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisor Professor Nobuyuki Matsui of University of Hyogo for the continuous support of my Ph.D. study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D. study.

I am deeply grateful to Associate Professor Teijiro Isokawa of University of Hyogo who helped me to share thoughts and idea of performing my research. I received generous supports and warm encouragement from him always. He is my best mentor and helped me with my everyday life as a Ph.D. student.

Special thanks are due to Professor Haruhiko Nishimura (University of Hyogo), Professor Masaki Kobayashi (University of Yamanashi), and Professor Kamiura Naotake (University of Hyogo) for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

Finally, I would like to thank to my family, and all of my friends for supporting my academic life.